VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Adam Kuklovský

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SDN PRO CLOUD COMPUTING
SDN FOR CLOUD COMPUTING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Adam Kuklovský**
AUTHOR

VEDOUCÍ PRÁCE — **Mgr. Karel Slavíček, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Adam Kuklovský ***ID:*** 203271
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

SDN pro cloud computing

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je analyzovat současné možnosti využití technologie SDN v oblasti cloud computingu a vytipovat k tomuto účelu vhodný SDN kontroler. Věcným výstupem práce je implementace tohoto kontroleru na jednom či více virtuálních serverech a vytvoření laboratorní úlohy, na které se budou moci další studenti seznámit s technologií SDN a jejím využití pro cloudové aplikace.

DOPORUČENÁ LITERATURA:

[1] SDN: Software Defined NEtworks [Book]. O'Reilly Media - Technology and Business Training [online] Copyright (c) 2019 Safari Books Online. [cit. 15.09.2019]. Dostupné z: https://www.oreilly.com/library/view/sdn-software-defined/9781449342425/

[2] GORANSSON, Paul, Cuck BLACK a Timothy CULVER, Software defined networks: a comprehensive approach. Second edition. Singapore: Morgan Kaufmann, [2017]. ISBN 9780128045558.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Mgr. Karel Slavíček, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

UPOZORNĚNÍ:
Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Bakalárska práca sa venuje technológii SDN. Práca obsahuje teoretickú časť a praktickú. V teoretickej časti je popísaný koncept SDN sietí, fungovanie protokolu OpenFlow, porovnanie jednotlivých open source SDN kontrolérov a reálne nasadenie technológie SDN. Praktická časť obsahuje postup inštalácií jednotlivých komponentov na vytvorenie topológie a následné realizovanie laboratórnej úlohy. Laboratórna úloha oboznámi študentov s vytváraním a modifikovaním tokov v SDN sieti.

KĽÚČOVÉ SLOVÁ

SDN, OpenDaylight, OpenFlow, Softvérovo-definované siete, Open vSwitch, OpenDaylight OpenFlow Manager,

ABSTRACT

The theme of this bachelor work is SDN technology. It consists of both practical and theoretical part. In theory section, we described the main concept of the SDN network, functioning of OpenFlow protocol, comparison of individual open source SDN controllers and real onset of SDN technology. Practical section contains procedure of individual components being installed to create topology and to forward realization of laboratory task. The laboratory work also depicts the way of creation and modification of streams in the SDN network.

KEYWORDS

SDN, OpenDaylight, OpenFlow, Software-defined networking, Open vSwitch, OpenDaylight OpenFlow Manager,

KUKLOVSKÝ, Adam. *SDN pro cloud computing*. Brno, Rok, 65 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Mgr. Karel Slavíček, Ph.D.

Vysadené pomocou balíčku `thesis` verzie 3.05; `http://latex.feec.vutbr.cz`

VYHLÁSENIE

Vyhlasujem, že svoju bakalársku prácu na tému „SDN pro cloud computing“ som vypracoval samostatne pod vedením vedúceho bakalárskej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce pánovi Mgr. Karel Slavíček, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Úvod **1**

1 Softvérovo-definované siete **2**
1.1 Dôvod vzniku Softvérovo-definovaných sietí . . . 2
1.2 Princíp fungovania Softvérovo-definovaných sietí . . . 2
1.2.1 Kontrolná rovina . . . 2
1.2.2 Dátová rovina . . . 3
1.3 Architektúra SDN . . . 3
1.3.1 Aplikačná vrstva . . . 3
1.3.2 Riadiaca vrstva . . . 4
1.3.3 Vrstva infraštruktúry . . . 4
1.4 OpenFlow . . . 5
1.4.1 OpenFlow kontrolér . . . 6
1.4.2 OpenFlow protokol . . . 6
1.4.3 Zabezpečený kanál . . . 6
1.4.4 Základy OpenFlow . . . 7
1.4.5 Porty a fronty na portoch . . . 7
1.4.6 Tabuľka tokov . . . 8
1.4.7 Porovnávanie paketov . . . 8
1.4.8 Akcie a posielanie paketov . . . 10
1.4.9 Správy medzi kontrolérom a prepínačom . . . 11
1.4.10 Programovanie tabuľky tokov . . . 14
1.4.11 Základné posielanie paketov . . . 15
1.4.12 Zasielanie paketov do kontroléra . . . 16
1.5 SDN kontroléry . . . 17
1.5.1 ONOS Open Network Operating System . . . 17
1.5.2 OpenDaylight . . . 18
1.5.3 Ryu . . . 18
1.5.4 Floodlight . . . 19
1.5.5 Maestro . . . 19
1.6 QoS v SDN sieti . . . 19
1.6.1 Vzťah medzi QoS a SDN . . . 20
1.7 Netconf . . . 21
1.7.1 Použitie Netconf u SDN . . . 22
1.7.2 Yang . . . 23
1.8 NFV: Virtualizácia sieťových funkcií . . . 24
1.8.1 Vztah medzi SDN a NFV . . . 25

1.9 Virtualizácia v oblasti dátových centier 25

2 Výsledky študentskej práce **27**

2.1 OpenDaylight 27
2.1.1 Koncepty a nástroje OpenDaylight 27
2.1.2 Clustering 27
2.1.3 Bezpečnosť 28
2.2 Open vSwitch 28
2.2.1 Mobilita 29
2.2.2 Reakcia na dynamiku siete 29
2.2.3 Podporované platformy 29
2.3 VMware ESXi 29
2.3.1 vSphere 30
2.4 Topológia siete 30
2.5 Hardvérové riešenie 31
2.6 Príprava prostredia 31
2.6.1 Ubuntu 18.04 31
2.6.2 Inštalácia OpenDaylight 31
2.6.3 Inštalácia OVS (Open Vswitch) 33
2.6.4 Inštalácia OpenDaylight features 33
2.6.5 OpenDaylight grafické rozhranie 35
2.6.6 Prepojenie OpenDaylight a OVS 35
2.7 Prvotné riešenie topológie 37
2.7.1 Inštalácia KVM 37
2.7.2 Použitie VM v KVM 38
2.7.3 Prepojenie KVM VM a OVS 38
2.8 Realizácia laboratórnej úlohy 39
2.8.1 Práca s ESXi vSwitch 39
2.8.2 Prepojenie OVS s VM 40
2.9 OpenDaylight OpenFlow Manager (OFM) 40
2.9.1 Úprava tokov cez OFM 42

Závěr **44**

Literatúra **45**

Zoznam symbolov, veličín a skratiek **49**

Zoznam príloh **51**

A Príklad tokov v JSON formáte **52**
A.1 Príklad toku Drop 52
A.2 Príklad toku ARP 52
A.3 Príklad toku pre povolenie ICMP 53

B Laboratórny návod **56**
B.1 Ciel úlohy 56
B.2 Pracovisko 56
B.3 Úlohy 56
B.4 Teoretický úvod 56
B.4.1 Dôvod vzniku Softvérovo-definovaných sietí 56
B.4.2 Princíp fungovania Softvérovo-definovaných sietí 57
B.4.3 Architektúra SDN 58
B.4.4 Open vSwitch 59
B.4.5 OpenDaylight 60
B.5 Postup riešenia 60
B.5.1 Topológia 60
B.5.2 Spustenie potrebných aplikácií 61
B.5.3 Úprava tokov 62
B.5.4 Samostatná práca 64
B.6 Otázky 65
B.7 Upratanie pracoviska 65

Zoznam obrázkov

1.1 SDN architektúra[5] 5
1.2 Kontrolér–Prepínač zabezpečený kanál [16] 8
1.3 Relácia protokolu Kontrolér-Prepínač [6] 13
1.4 Programovanie tabuľky tokov [20] 15
1.5 Posielanie paketov [21] 16
1.6 QoS [8] 20
1.7 OpenFlow vs Netconf 23
2.1 Topológia laboratórnej úlohy. 30
2.2 Spustenie OpenDaylight 32
2.3 Prihlásenie do grafického rozhrania. 35
2.4 Kontrola prepojenia OpenDaylight a OVS v grafickom rozhraní DLUX. 36
2.5 Kontrola prepojenia OpenDaylight a OVS v príkazovom riadku OVS. 36
2.6 Kontrola podpory KVM. 37
2.7 KVM grafické rozhranie. 38
2.8 Topológia realizovaná v ESXi servery. 39
2.9 Realizovaná topológia zobrazená v grafickom rozhraní OpenDaylight. 40
2.10 Grafické rozhranie OFM. 41
2.11 Výpis tokov v OVS. 43
B.1 SDN architektúra 59
B.2 Topológia laboratórnej úlohy. 60
B.3 Nastavenie OVS 61
B.4 Spustenie OpenDaylight 62
B.5 Kontrola prepojenia OpenDaylight a OVS v grafickom rozhraní DLUX. 63
B.6 Nastavenie toku pre zahadzovanie komunikácie. 64

Zoznam tabuliek

1.1 Typy OFTP správ v OpenFlow [4] 12
2.1 Nastavenie pre toky Drop a ARP. 42
2.2 Nastavenie pre ICMP toky. 43
B.1 Nastavenie ICMP tokov. 64
B.2 Tabuľka Ethernet type. 65
B.3 Tabuľka IP protokolov. 65
B.4 Tabuľka ICMP typov. 65

Úvod

Vďaka rýchlemu prijatiu prostredí clud compuntingu na hosťovanie rôznych aplikácií, ako sú web, virtuálna realita, vedecké výpočty a veľké dáta, sa potreba poskytovania cloudových služieb zo zárukami kvality služieb (QoS) stáva kritickou. Pre správu cloudových dátových centier je dôležité zabrániť porušovaniu dohody o úrovni služieb (SLA) a udržiavať QoS v heterogénnych prostrediach s rôznymi aplikačnými kontextmi. Preto sa zameriava na optimalizáciu latencie služieb a energetickej efektívnosti dynamicky s cieľom využívať výhody nájomcov aj poskytovateľov cloud computingu.

Virtuálne zariadenia (VM) sú jedna z hlavných virtualizačných technológií pre hosťovanie cloudových služieb, môžu zdielať počítačové a sieťové zdroje. Na zmiernenie porušení SLA a garantovaniu QoS je potrebné neustále optimalizovať umiestňovanie virtuálnych počítačov v dynamickom prostredí. Živá migrácia VM je kľúčovou technológiou na premiestnenie spustených virtuálnych počítačov medzi fyzickými hostiteľmi bez toho, aby to narušila dostupnosť. V dátových centrách s povoleným SDN je živá VM migrácia dynamickým nástroj riadenia obsahujúci rôzne ciele plánovania zdrojov, ako je napríklad vyrovnávanie záťaže, rezervácie zdrojov, energetická úspora, stratégia ukladania, odolnosť voči chybám, plánovaná údržba, ako aj evakuácia VM do iných dátových stredísk pred udalosťami ako je zemetrasenie a záplavy ktoré si vyžadujú prispôsobenie polohy VM[1].

Vecný výstup tejto práce bude laboratórna úloha, na ktorej sa budú študenti oboznamovať s technológiou softvérovo-definovaných sietí v oblasti cloud computingu.

1 Softvérovo-definované siete

1.1 Dôvod vzniku Softvérovo-definovaných sietí

Postupom času ako sa začal zvyšovať výkon zariadení a začali sa využívať cloudové služby, to viedlo k logickému zvyšovaniu nárokov na počítačové siete. Ako na ich komplexnosť tak aj veľkosť prenášaných dát. Problém nastáva pokiaľ klasickú sieť chceme rozšíriť o nové zariadenie. Tento proces býva časovo dosť náročný z pohľadu konfigurácie zariadenia a troubleshutingu danej konfigurácie. Aby sme tomuto problému predišli, začala sa riadiaca časť siete zjednocovať na jedno zariadenie.

Technológia, ktorá toto zabezpečuje sa nazýva SDN. Príchodom SDN prišli nové inovácie ktoré, uľahčujú spravu a konfiguráciu siete. Jedna z inovácií je rozdelenie riadiacej a dátovej roviny sieťových zariadení. Týmto rozdelením sa dosiahli to, že o spravu a kontrolu siete sa stará centrálna jednotka ktorá, sa nazýva kontrolér. SDN našlo uplatnenie ako v centralizácií a virtuálizácií lokálnych sietí tak aj v oblasti cloud computing [2].

1.2 Princíp fungovania Softvérovo-definovaných sietí

Jednou z myšlienok využitia SND je v oddelení riadiacích a dátových rovinách sieťových zariadení. Takéto oddelenie dáva prevádzkovateľovi siete určité výhody, pokiaľ ide o centralizované alebo semi-centralizované distribučné možnosti. Tiež má potenciálnu ekonomickú výhodu založenú na schopnosti zjednotiť sa na jednom alebo viacerých miestach, čo umožňuje jednoduchšiu konfiguráciu, čo v konečnom dôsledku vychádza lacnejšie ako klasický spôsob.

Oddelenie kontrolnej a dátovej roviny je jedna zo základných vlastností SDN [3].

1.2.1 Kontrolná rovina

Kontrolná rovina zaradí lokálne dáta, ktoré použije na vytvorenie vstupov v smerovacej tabuľke, ktoré využije dátová rovina na smerovanie toku dát na vstupné alebo výstupné porty na zariadení. Súbor, do ktorého sa ukladajú informácie o topológii siete sa nazýva „routing information base“ (RIB). Táto databáza sa udržuje konzistentná (bez slučiek) pomocou výmeny informácií medzi jednotlivými kontrolnými rovinami v sieti. Vstupy smerovacej tabuľky sa nazývajú „forwarding information base“ (FIB) a zrkadlia sa medzi kontrolnú a dátovú rovinu zariadenia. FIB sa naprogramuje, keď sa RIB považuje za konzistentný a nemenný. Na vykonanie tejto úlohy musí riadiaca jednotka vytvoriť pohľad na sieťovú topológiu, ktorá spĺňa určité obmedzenia. Tento pohľad na sieť sa dá naprogramovať ručne, naučiť na základe

pozorovania siete alebo vytvoriť na základe zozbieraných informácií prostredníctvom komunikácie s inými príkladmi kontrolných rovín, čo môže byť prostredníctvom použitia jedného alebo viacerých smerovacích protokolov, manuálneho programovania alebo kombináciou oboch.

Ak je prijatý paket, ktorý prišiel z neznámej MAC adresy, je pozastavený alebo presmerovaný na riadiacu rovinu zariadenia, kde sa ju zariadenie naučí, spracuje a pošle ďalej. Rovnaké správanie je zachované, aj keď sa jedná o riadenie, ako sú správy o smerovacom protokole (napr. OSPF link-state oznámenie). Po doručení paketu do kontrolnej roviny sa spracujú informácie obsiahnuté v protokole. Tieto informácie môžu mať za následok zmenu v RIB, ako aj prenos dodatočných správ upozorňujúcich na túto aktualizáciu(t.j nová trasa je naučená). Keď sa RIB stane stabilnou, FIB sa aktualizuje v kontrolnej aj dátovej rovine. Následne bude aktualizované zasielanie údajov a budú odrážať zmenu. Kedže paket bol jednou z neurčitých MAC adries, riadiaca rovina vráti paket do dátovej roviny, ktorá zodpovedajúcim spôsobom odošle paket [11].

1.2.2 Dátová rovina

Dátová rovina spracováva prichádzajúce datagramy (na drôte, optickom vlákne alebo bezdrôtovo) prostredníctvom série operácií na úrovni pripojení, ktoré tieto datagramy zachytávajú a vykonávajú základnú kontrolu. Správny datagram sa spracováva v dátovej rovine prostredníctvom vyhľadávaní v tabuľke FIB, ktorá je naprogramovaná skôr kontrolnou rovinou. Toto sa niekedy označuje, ako rýchla cesta spracovávania paketov, pretože nepotrebujú už žiadné ďalšie dodatočné informácie, okrem tých, ktoré sú obsiahnuté vo FIB tabuľke. Výnimka nastáva, keď paket nesplňuje tieto pravidlá, ako napríklad, keď sa zistí neznámy cieľ, tak sa tieto dáta odošlú do procesora trasy, kde ich kontrolná rovina spracuje s použitím RIB tabuľky [12].

1.3 Architektúra SDN

SDN sa vo všeobecnosti skladá z troch vrstiev:

- Aplikačná vrstva
- Riadiaca vrstva
- Vrstva infraštruktúry

1.3.1 Aplikačná vrstva

Otvorená oblasť, ktorá umožňuje vývoj inovatívnych aplikácií s využívaním všetkých informácií o sieťovej topológii. Je niekoľko druhov aplikácií, ktoré je možno vyvíjať

zo zameraním na riešenie problémov so sieťou, sieťová politika, bezpečnosť, aplikácie súvisiace s automatizáciou siete, sieťová konfigurácia a správa a monitorovanie siete. Tieto aplikácie môžu ponúknuť rôzne komplexné riešenia pre podnikové siete a siete dátových centier v reálnom čase. Sieťový sprostredkovatelia ponúkajú vlastné riešenia svojich SDN aplikácií[5].

1.3.2 Riadiaca vrstva

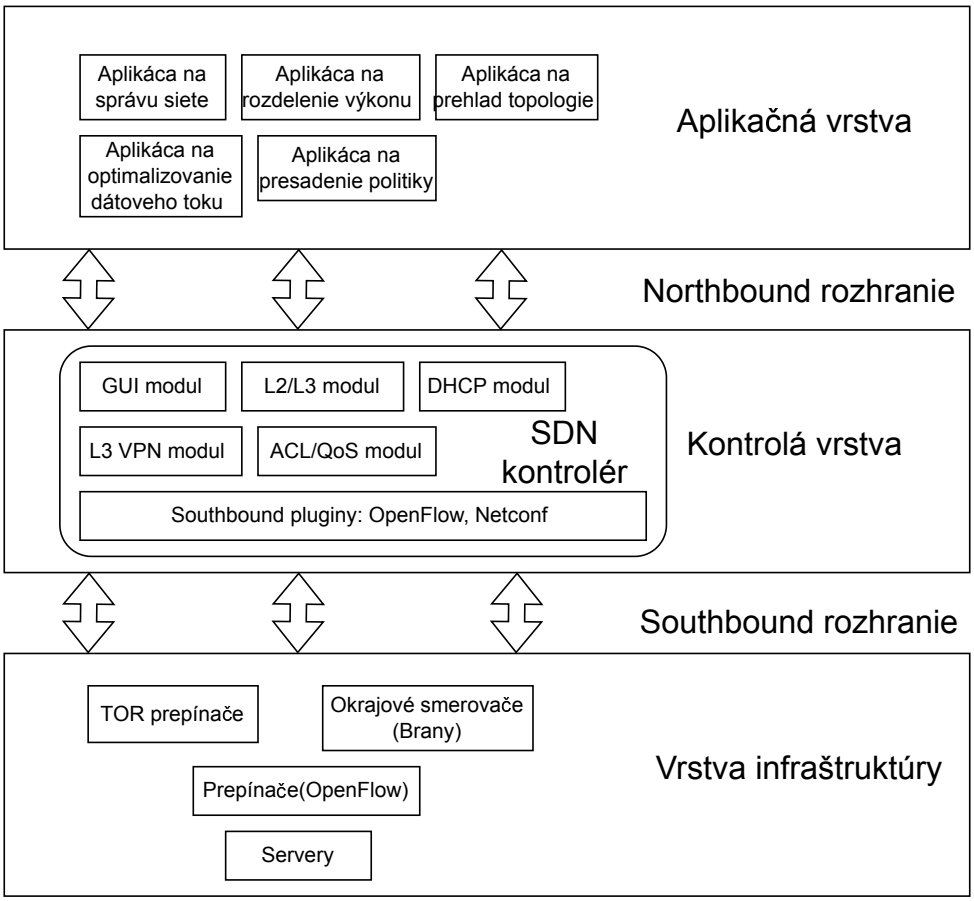
Je miesto, kde sa uplatňuje kontrolná rovina a kontrolu nad sieťou zabezpečuje SDN kontrolér. Toto je oblasť, kde každý sieťový sprostredkovateľ pracuje na tom, aby prišiel s vlastnými produktami pre SDN kontrolér alebo framework. V tejto vrstve je v kontroléri napísaných veľa administratívnych logík, aby bolo možné udržiavať rôzne typy informácií o sieti, podrobnosti o stave, podrobnosti o topológii, štatistické údaje a ďalšie.

Pretože SDN kontrolér slúži na správu siete, tak musí mať riadiacu logiku pre prípady použitia v reálnom svete, ako je prepínanie, smerovanie, L2 VPN, L3 VPN, bezpečnostné pravidlá pre firewall, DNS a DHCP. Niekoľko sieťových sprostredkovateľov a komunít s otvoreným zdrojovým kódom pacujú na implementácií týchto prípadov používania na ich SDN kontrolér. Implementácia týchto služieb sa uskutočňuje v API (aplikačná vrstva), čo uľahčuje implementáciu správcom, ktorí potom pomocou aplikácie na SDN kontroléri konfigurujú, spravujú a monitorujú základnú sieť. Kontrolná rovina sa nachádza v strede a oddeľuje dva typy rozhraní[5]:

- **Southbound rozhranie** : je určený na komunikáciu s nižšou vrstvou infraštruktúry sieťových prvkov a vo všeobecnosti je realizovaný pomocou southbound protokolov - OpenFlow, Netconf, Ovsdb...
- **Northbound rozhranie** : je určený na komunikáciu s hornou aplikačnou vrstvou a vo všeobecnosti je realizovaný prostredníctvom REST API SDN kontroléra.

1.3.3 Vrstva infraštruktúry

Skladá sa z rôznych sieťových zariadení, ktoré tvoria základnú sieť na smerovanie. Môže to byť sada prepínačov a smerovačov v dátových centrách. Táto vrstva je fyzická, na ktorú sa prostredníctvom riadiacej vrstvy stanovila virtualizácia siete(kde SDN kontrolér riadi základnú fyzickú sieť) [5].



Obr. 1.1: SDN architektúra[5]

1.4 OpenFlow

OpenFlow je samostatná podskupina technológií zahrnutých do veľkého okruhu ktoré spadajú pod SDN. OpenFlow je dôležitý nástroj a katalyzátor inovácií, ktorý definuje komunikačný protokol medzi dátovou rovinou SDN a kontrolnou rovinou SDN, ako aj časť správania sa dátovej roviny. Nepopisuje spávanie samotného kontroléra. Zatiaľ čo existujú rôzne prístupy k SDN, OpenFlow je jediný neproprietárny univerzálny protokol na programovanie smerovacej roviny na SDN prepínačoch.

OpenFlow pozostáva z kontroléra OpenFlow, ktorý komunikuje s jedným alebo viacerými OpenFlow prepínačmi. OpenFlow definuje konkrétne správy a formáty správy, ktoré sa menia medzi kontrolérom (riadiaca rovina) a zariadením (dátová rovina). Správanie OpenFlow špecifikuje, ako by malo zariadenie reagovať v rôznych situáciach a ako by malo reagovať na príkazy z ovládača[13].

1.4.1 OpenFlow kontrolér

Riadiaca rovina OpenFlow sa líši od pôvodnej riadiacej roviny tromi kľúčovými spôsobmi.

1. Môže naprogramovať rôzne prvky dátovej roviny so spoločným štandardným jazykom OpenFlow.
2. Existuje na samotnom hardwervom zariadení ako preposielacia rovina, na rozdiel od tradičných prepínačov, kde sú riadiaca a dátová rovina na rovnakom fyzickom zariadení. Toto oddelenie je možné, pretože kontrolér môže programovať prvky dátovej roviny na diaľku cez internet.
3. Riadiaca jednotka môže naprogramovať viac prvkov dátovej roviny z jednej inštalácie riadiacej roviny.

Riadiaca jednotka OpenFlow je zodpovedná za programovanie všetkých pravidiel na kontrolovanie a preposielanie paketov na prepínači. Zatiaľ čo tradičný smerovač by vykonával smerovacie algoritmy na určenie toho, ako bude fungovať preposielacia tabuľka. Túto funkciu alebo jej ekvivalentnú dokáže nahradiť riadiaca jednotka. Akékoľvek zmeny, ktoré vedú k prepočítaniu trás bude naprogramované pomocou kontroléra[14].

1.4.2 OpenFlow protokol

OpenFlow protokol definuje komunikáciu medzi OpenFlow kontrolérom a zariadením. Protokol je to, čo presne definuje OpenFlow technológiu. Vo vlastnej podstate, protokol sa skladá zo súboru správ, ktoré sa odosielajú z kontroléra na zariadenie. Správy umožnujú kontroléru programovať zariadenie tak, aby umožnil starostlivú kontrolu pred spracovaním užívateľského toku. Tok je definovaný ako množina paketov z jedného konca siete na druhý koniec siete. Koncové body môžu byt definované ako IP TCP/UDP porty, koncové body VLAN, koncové bod tunela na tretej úrovni.

Jedna sada pravidiel popisuje akcie preposielania, ktoré by malo zariadenie vykonať pre všetky pakety patriace do toho toku, keď ovládač definuje tok, poskytuje prepínaču informácie, ktoré potrebuje vedieť ako zaobchádzať s prichádzajúcimi paketmi, ktoré zodpovedajú tomuto toku.

1.4.3 Zabezpečený kanál

Zabezpečený kanál je cesta používaná na komunikáciu medzi ovládačom OpenFlow a zariadením Openflow. Všeobecne je táto komunikácia zabezpečená asymetrikým šifrovaním založeným na TLS, aj keď sú povolené nešifrované spojenia typu TCP. Tieto spojenia môžu byť v pásme alebo mimo pásma. V prípade mimo pásma spojenie zabezpečeného kanála vstupuje do prepínača cez port, ktorý nieje prepínaný

dátovou rovinou. Niektoré staršie sieťové zásobníky doručia správy OpenFlow prostredníctvom procesu zabezpečeného kanála a zabezpečeného kanála v prepínači, kde sa všetky OpenFlow správy analyzujú a spracovávajú. Preto je zabezpečený kanál mimo pásma relevantný iba v prípade hybridného prepínača OpenFlow.

Na Obr. 1.2 vidíme ako prichádza OpenFlow správa z kotroléra cez port K, ktorý je súčasťou dátovej roviny OpenFlow. V tomto prípade bude s týmito paketmi manipulovať logika porovnávania paketov OpenFlow znázornená na Obr. 1.2. Tabuľky sledujúce tok údajov budú skonštruované tak, aby sa táto OpenFlow prevádzka bola presmerovaná na virtuálny port LOCAL (Port LOKAL sa používa, keď sú spravy OpenFlow z kontroléra prijímané na porte, ktorý prijíma pakety prepínane dátovou rovinou OpenFlow. LOKAL označuje že paket musí byť spracovaný miestnym riadiacim softvérom OpenFlow.), čo vedie k odovzdávaniu správ do procesu bezpečného kanálu.

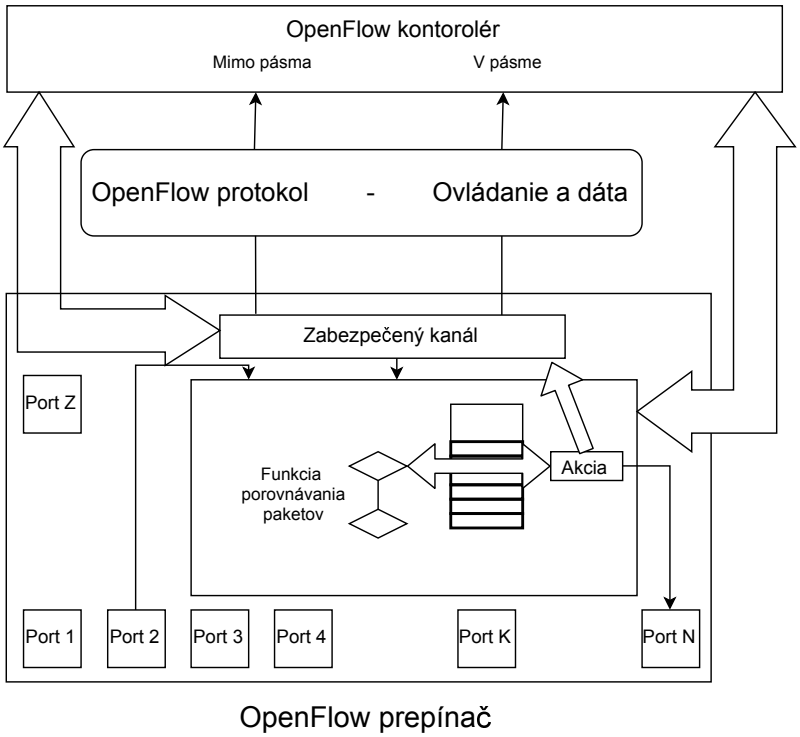
Keď je kotrolér a všetky prepínače, ktoré riadi, umiestnené v prísne kontrolovanom prostredí, ako je napríklad dátové centrum, môže byť rozumné zvážiť použitie zabezpečenie kanála na základe šifrovania TLS. Dôvod je taký, že pri požití tohto druhu bezpečnosti vznikajú režijné náklady. Ak toto nieje potrebné, je lepšie sa týmto nákladom vyhnúť. Aby sa dalo TLS uviesť do praxe, je potrebné získať a nakonfigurovať bezpečnostné certifikáty na jednotlivé zariadenia. Implementácia na jednotlivé zariadenia môže byt časovo náročná, alebo chybová pre niekoho, kto nie je oboznámený s touto metódou.

1.4.4 Základy OpenFlow

Základný princíp fungovania a základných komponentov OpneFlow bude popísaný vo verzií OpenFlow 1.0. Verzia OpenFlow je prvá verzia z ktorej vychádzajú ďalšie verzie: 1.1, 1.2, 1.3, 1.4 a 1.5.

1.4.5 Porty a fronty na portoch

Špecifikácia OpenFlow definuje koncepciu *OpenFlow Port*. Vo verzií OpenFlow 1.0 OpenFlow Port zodpovedá fyzickému portu. Tento koncept sa využíva aj v iných verziách OpenFlow. Sofistikované prepínače už mnoho rokov podporujú viacej front na fyzický port. Tieto fronty sa uschovávajú podľa plánovacích algoritmov, ktoré umožňujú poskytovanie rôznych úrovní kvality služieb (QoS) pre rôzne typy paketov. OpenFlow zahŕňa tento koncept a umožňuje mapovanie toku na už definovanú frontu na výstupnom porte. Podpora QoS je vo verzií OpenFlow dosť základná. Podpora QoS sa postupom verzií výrazne rozšírila[17].



Obr. 1.2: Kontrolér–Prepínač zabezpečený kanál [16]

1.4.6 Tabuľka tokov

Tabuľka tokov leží v jadre definície prepínača OpenFlow. Tabuľka tokov pozostáva zo vstupov toku. Zoznam týchto tokov obsahuje polia záhlavia, počítadlá a akcie spojené s touto položkou. Polia záhlavia sa používajú ako kritéria zhody na určenie, či predchádzajúci paket zodpovedá tejto položke. Ak existuje zhoda, paket patrí do tohto toku. Počítadlá sa používajú na sledovanie štatistík týkajúcich sa toho toku, napríklad koľko paketov sa preposlalo alebo vyhodilo v tomto toku. Polia akcií popisujú, čo by mal prepínač robiť s paketom, zodpovedajúcemu tejto položke[17].

1.4.7 Porovnávanie paketov

Keď paket príde k OpenFlow prepínaču zo vstupného portu(v niektorých prípadoch z ovládača), porovná sa s tabuľkou toku, aby sa zistilo, či existuje záznam zodpovedajúceho toku. Nasledujúce polia zhody priradené k prichádzajúcemu paketu sa môžu na porovnanie zo vstupmi toku:

- Vstupný port prepínača
- VLAN ID
- Vlan priorita
- Ethernet zdrojová adresa
- Ethernet cieľová adresa

- Typ etherent rámca
- Zdrojová IP adresa
- cieľová IP adresa
- IP protokol
- TCP/UDP zdrojový port
- TCP/UDP cieľový port
- Typ služby (ToS) bity

Týchto dvanásť zápisových polí sa súhrnne označujú ako základné polia zhody. Polia zhody záznamu toku môžu byť označené pomocou bitovej masky, čo znamená, že akákoľvek hodnota, ktorá sa zhodujú odmaskované bity v poliach zhody prichádzajúceho paketu bude zhoda. Záznamy toku sa spracovávajú v poradí, a keď sa objaví zhoda, sa pri tejto tabuľke tokov nevykonávajú ďalšie pokusy o zápis. Z tohto dôvodu je možné, aby existovalo viac záznamov zodpovedajúcich tokov pre paket, ktorý ma byť prítomný v tabuľke tokov. Zmysluplný je iba prvý záznam toku, ktorý sa zhoduje. Ostatné pakety sa po prvej zhode nebudú považovať za relevantné.

Verzia OpenFlow 1.0 nehovorí o tom, ktoré z týchto dvanástich polí sú povinné a ktoré sú voliteľné. ONF objasnil tento zmätok definovaním troch rôznych typov zhody vo svojom programe testovania zhody. Tieto tri úrovne sú:

- Úplná zhoda, čo znamená, že je podporovaných všetkých dvanásť porovnávacích polí.
- Zhoda druhej vrstvy, je porovnaná zhoda len hlavičky druhej vrstvy
- Zhoda tretej vrstvy, je porovnaná zhoda len hlavičky tretej vrstvy

Ak sa dosiahne koniec v tabuľke tokov bez nájdenia zhody, nazýva sa to miss tabuľka. V prípade, že tabuľka chýba, je paket poslaný do riadiacej jednotky.(Toto neplatí už v novších verziách). Ak sa nájde zodpovedajúci záznam tokov, akcie spojené s týmto spojené s týmto prúdom toku určia, ako sa bude s paketom zaobchádzať. Najzákladnejšou akciou predpísanou OpenFlow prepínačom je, ako poslať tento paket.

Je dôležité si uvedomiť, že táto funkcia porovnávania paketov bola navrhnutá ako abstrakcia k tomu, ako dnes funguje hardvér prepínačov v súčastnosti. Prvé verzie OpenFlow boli navrhnuté tak, aby špecifikovali správanie pri preposielaní existujúcich komerčných prepínačov prostredníctvom tejto abstrakcie. Dobrá abstrakcia skryje podrobnosti o veciach, ktoré boli vytvorené abstrakciou, pričom stále umožňujú dostatočne jemnú kontrolu na vykonanie potrebných úloh. V neskorších verziách OpenFlow sa pridáva bohatšia funkčnosť. Špecifikácia prevyšuje realitu dnešného hardvéru. V tomto bode už neposkytuje čistú abstrakciu súčasným implementáciám , ale špecifikuje správanie pri prepínaní hardvéru, ktorý ešte musí byť postavený[18].

1.4.8 Akcie a posielanie paketov

Požadované akcie, ktoré musia byť podporované vstupom toku, sú buď na výstup alebo na zahodenie zhodného paketu. Najčastejším prípadom je, že výstupná akcia určuje fyzický port, na ktorý by mal byť paket ďalej poslaný. Vo verzií OpenFlow 1.0 je definovaných päť špeciálnych virtuálnych portov ktoré majú osobitný význam pre výstupnú akciu. Sú to **LOCAL, ALL, CONTROLLER, IN_PORT a TABLE**.

LOKAL diktuje, že paket by mal byť preposlaný do lokálneho ovládacieho softvéru OpenFlow. LOCAL sa používa, keď sú OpenFlow správy prijímané z kotroléra na porte, ktorý prijíma pakety prepínane dátovou rovinou OpenFlow. LOKAL označuje že, paket musí byť spracovaný miestnym riadiacim softvérom OpenFlow.

ALL sa používa na zaplavenie paketov zo všetkých portov okrem vstupného portu. Toto poskytuje možnosť základného broadcastu na OpenFlow prepínači.

CONTROLLER označuje, že prepínač by mal tento paket ďalej do OpenFlow kontroléra.

IN_PORT dáva prepínaču pokyn, aby poslal paket späť z portu, ma ktorý prišiel. IN_PORT normálne vytvára situáciu spätnej slučky, čo by mohlo byť užitočné pre určité prípady. Jedným z možných scenárov je prípad bezdrôtového portu 802.11. V tomto prípade je normálne prijímať paket z tohto portu od jedného hosta a posielať ho druhému hostovy cez ten istý port. Toto je potrebné urobiť veľmi opatrne, aby sa nevytvorili neúmyselné situácie spätnej slučky. Protokol preto vyžaduje výslovné stanovenie tohto zámeru prostredníctvom špeciálneho virtuálneho portu.

Ďalším prikladom, kde sa využíva IN_PORT, je Edge Virtual Bridging (EVB). Edge Virtual Bridging definuje službu reflexného prenosu medzi fyzickým prepínačom v dátovom centre a ľahkým virtuálnym prepínačom v rámci servera, ktorý je známy ako agregátor portov Virtual Edge Port (VEPA). Štandardný most IEEE 802.1Q na okraji siete bude odrážať pakety späť z portu, na ktorý prichádzajú, aby umožnil dvom virtuálnym počítačom na rovnakom serveri komunikovať medzi sebou. Táto reflexná služba môže byť podporovaná cieľovým IN_PORTom v OpenFlow pravidlách.

TABLE je virtuálny port, ktorý sa vzťahuje iba na pakety, ktoré riadiaca jednotka prepne na prepínač. Takéto pakety prichádzajú ako súčasť správy PACKET_OUT z kontroléra, čo zahŕňa zoznam akcií. Tento zoznam akcií bude spravidla obsahovať výstupnú akciu, ktorá špecifikuje číslo portu. Ovládač môže chcieť priamo špecifikovať výstupný port pre tento dátový paket, alebo ak si želá aby bol výstupný port určený normálnym potrubím na spracovanie paketov OpenFlow, môže to urobiť aj zadaním TABLE ako výstupného portu.

Existujú ešte ďalšie dva virtuálne porty, ich podpora je však vo verzií 1.0 vo-

liteľná. Prvý z nich je **NORMAL** virtuálny port. Keď výstupná akcia preposiela paket na virtuálny port NORMAL, prepíše paket do staršej logiky preposielania prepínača. Kontrastuje to s virtuálnym portom LOCAL, ktorý označuje, že paket má odovzdať miestnemu spracovaniu riadenia OpenFlow. Naopak pakety,ktorých pravidlo zhody označuje NORMAL ako výstupný port, zostanú v ASIC, aby sa dali vyhľadať v ďalších tabuľkách preposielania, ktoré sú obsadené miestnou riadiacou rovinou (nonOpenFlow). Použitie NORMAL ma zmysel iba v prípade hybridného prepínača.

Zostávajúci virtuálny port je **FLOOD**. V tomto prípade prepínač odošle kópiu paketu zo všetkých portov okrem vstupného portu.

Vo verzií OpenFlow 1.0 sú dve voliteľné akcie. Sú to enqueue a modifik-field. Enqueue akcia vyberie špecifický front patriaci konkrétnemu portu. To by sa použilo v spojení s výstupnou akciou a používa sa na dosiahnutie požadovaných úrovní QoS prostredníctvom použitia viacerých frontov priority na porte. Nakoniec akcia modify-field informuje prepínač, ako upraviť určité polia záhlavia. Špecifikácia obsahuje zdĺhavý zoznam polí, ktoré sa môžu touto akciou zmeniť. Môžu sa meniť najmä hlavičky VLAN, zdrojová a cieľová adresa Ethernet, zdrojová a cieľová IP adresa a TTL pole. Pole modify je nevyhnutné pre najzákladnejšie smerovacie funkcie. Aby smerovali pakety tretej vrstvy, musí smerovač dekrementovať pole TTL pred poslaním paketu z určeného portu.

Ak sú s takouto položkou spojené viaceré akcie, objavia sa v zozname akcií, rovnako ako správa PACKET_OUT. Prepínač musí vykonať akcie v poradí, v akom sa nachádzajú v zozname akcií[19].

1.4.9 Správy medzi kontrolérom a prepínačom

Správy medzi prepínačom a kontrolérom sa prenášajú zabezpečeným kanálom. Tento bezpečný kanál je implementovaný prostredníctvom počiatočného pripojenia TLS cez TCP. Ak prepínač pozná IP adresu ovládača, prepínač začne spojenie, v ktorom kontrolér začne ovládať prepínač. Každá správa medzi kontrolérom a prepínačmi sa začína OpenFlow hlavičkou. Táto hlavička určuje číslo verzie OpenFlow, typ správy, dĺžku správy a ID transakcie správy. Rôzne typy správ vo verzií OpenFlow 1.0 sa nachádzajú v tab. 1.1. Správy spadajú do troch všeobecných kategórií: Symmetric, Controller-Switch a Async. Obr. 1.3 ukazuje najdôležitejšie z týchto správ v normálnom kontexte a ukazuje, kde sa zvyčajne používajú počas inicializácie, prevádzkovej alebo monitorovacej fáze dialógu kontrelér-spínač. Fázy monitorovania prevádzky sa vo všeobecnosti prekrývajú, ale kvôli prehľadnosti sú na obr. 1.3 zobrazené ako nespojité.

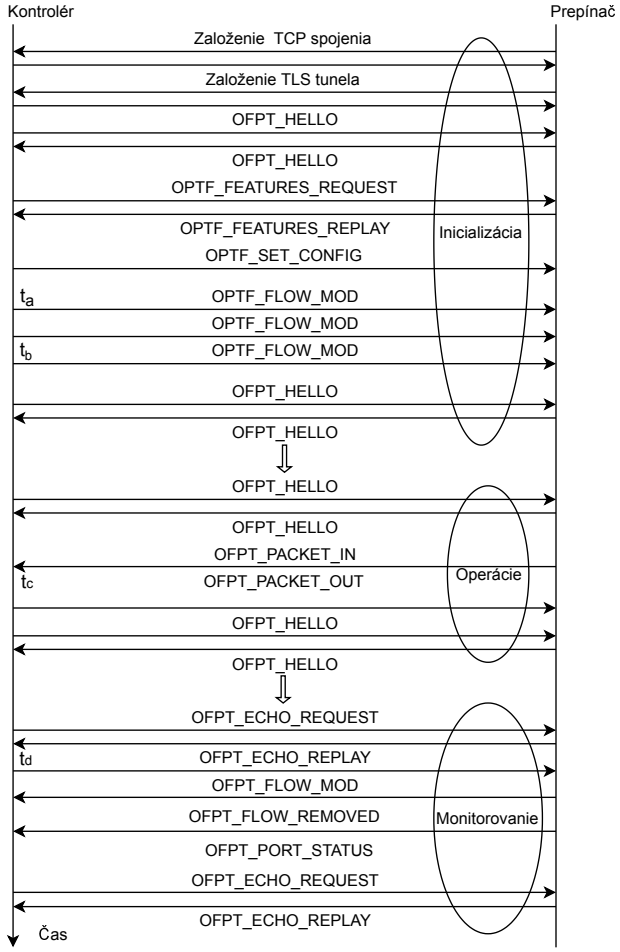
Symetrické správy môžu byť odosielané buď kontrolérom alebo prepínačom, bez

Tab. 1.1: Typy OFTP správ v OpenFlow [4]

Typ správy	Kategória	Podkategória
HELLO	Symmetric	Immutable
ECHO_REQUEST	Symmetric	Immutable
ECHO_REPLAY	Symmetric	Immutable
VENDOR	Symmetric	Immutable
FEATURES_REQUEST	Controller-Switch	Switch Configuration
FEATURES_REPLAY	Controller-Switch	Switch Configuration
GET_CONFIG_REQUEST	Controller-Switch	Switch Configuration
GET_CONFIG_REPLAY	Controller-Switch	Switch Configuration
SET_CONFIG	Controller-Switch	Switch Configuration
PACKET_IN	Async	NA
FLOW_REMOVED	Async	NA
PORT_STATUS	Async	NA
ERROR	Async	NA
PACKET_OUT	Controller-Switch	Cmd from controller
FLOW_MOD	Controller-Switch	Cmd from controller
PORT_MOD	Controller-Switch	Cmd from controller
STATUS_REQUEST	Controller-Switch	Statistics
STATUS_REPLAY	Controller-Switch	Statistics
BARRIER_REQUEST	Controller-Switch	Barrier
BARRIER_REPLAY	Controller-Switch	Barrier
QUEUE_GET_CONFIG_REQUEST	Controller-Switch	Queue configuration
QUEUE_GET_CONFIG_REPLY	Controller-Switch	Queue configuration

toho, aby boli druhou stranou vyžiadané. Správy HELLO sa vymieňajú po vytvorení zabezpečeného kanála, aby sa určilo najvyššie číslo verzie OpenFlow podporované zariadeniami. Protokol špecifikuje, že spodná z týchto dvoch verzií sa má použiť na komunikáciu Controller-Switch v tejto inštancii zabezpečeného kanála. Správy ECHO používajú obe strany počas života kanála na zaistenie, či je spojenie stále nažive a na meranie aktuálnej latencie alebo šírky pásma pripojenia. Správy VENDOR sú k dispozícii pre experimentálne vylepšenia špecifické pre dodávateľa.

Async správy sa odosielajú z prepínača do kontroléra bez toho, aby ich vyžadoval kontrolér. Správa PACKET_IN je spôsob, akým prepínač odovzdáva dátové pakety späť do kontroléra na spracovanie výnimiek. Prevádzka kontrolnej roviny sa zvyčajne prostredníctvom tohto hlásenia pošle späť do riadiacej jednotky. Prepínač môže informovať kontrolér o tom, že záznam toku je odstránený z tabuľky



Obr. 1.3: Relácia protokolu Kontrolér-Prepínač [6]

toku prostredníctvom správy FLOW_REMOVED. PORT_STATUS sa používa na komunikáciu o zmenách stavu portu, či už priamym zásahom používateľa alebo fyzickou zmenou samotného komunikačného média. Nakoniec prepínač použije správu ERROR na upozornenie kontroléra na problémy.

Controller-Switch je najširšia kategória správ Openflow. Ako je uvedené v tab. 1.1, tak ich môžeme rozdeliť do piatich podkategórií: *Switch Configuration, Command from Controller, Statistic, Queue Configuration* a *Barrier.* Správy konfigurácie prepínača pozostávajú z jednosmernej konfiguračnej správy a dvoch párov správ požiadavka–odpoveď. Jednosmerná správa SET_CONFIG používa kontrolér na nastavenie konfiguračných parametrov prepínača. Na obr. 1.3 je správa SET_CONFIG zaslaná počas inicializačnej fázy dialógu kontrolér-prepínač. Podobne sa správa pár

GET_CONFIG, používa na načítanie konfigurácie prepínača.

V kategórii Comand From Controller sú tri správy. PACKET_OUT je analógom už uvedeného PACKET_IN. Používa ju riadiaca jednotka na posielanie dátových paketov do prepínača na posielanie ďalej cez dátovú rovinu. Kontrolér modifikuje existjúce vstupy toku v prepínači prostredníctvom správy FLOW_MOD. PORT_MOD sa používa na zmenu stavu portu OpenFlow.

Statistic získava kontrolér z prepínača prostredníctvom dvojice správ STATS. Dvojica správ BARRIER používa kontrolér, aby sa zabezpečilo, že konkrétny príkaz OpenFlow z kontroléra sa realizuje na prepínači. Prepínač musí dokončiť vykonanie všetkých príkazov prijatých pred BARRIER_REQUEST skôr, ako vykonaná akýchkoľvek príkazov prijatých po ňom, a upozorní kontrolér, že dokončil tieto predchádzajúce príkazy prostredníctvom správy BARRIER_REPLAY odoslané naspäť do kontroléra.

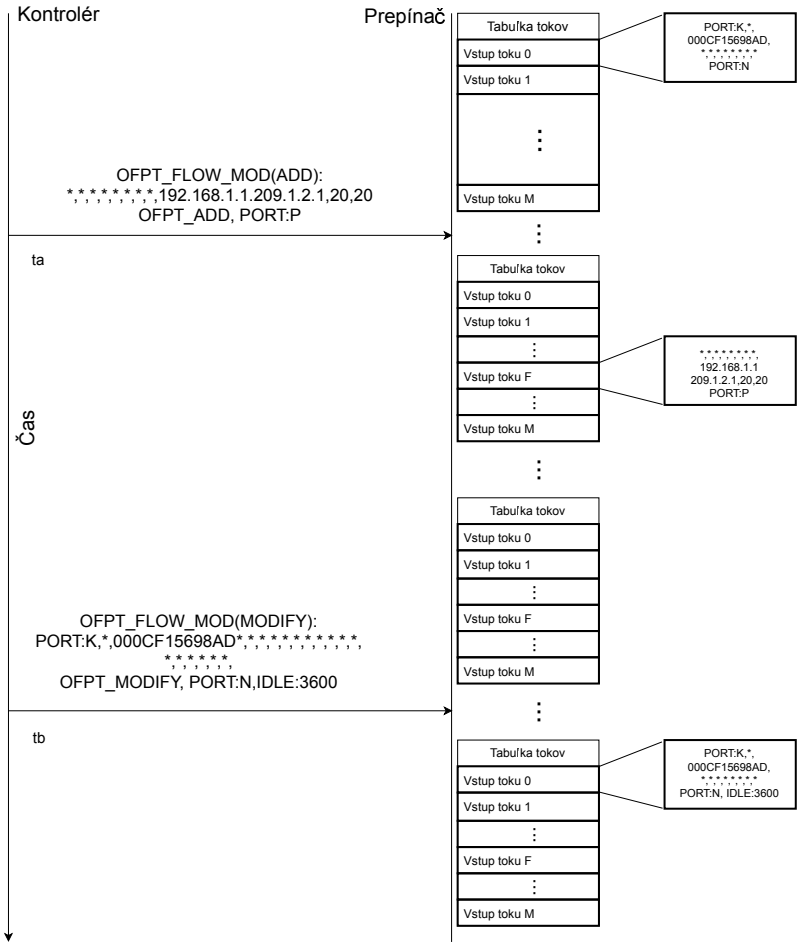
Dvojica správ queue configuration je trochu mylná v tom, že skutočná konfigurácia fronty presahuje rámec špecifikácie OpenFlow a očakáva sa, že sa uskutoční pomocou nešpecifikovaného mechanizmu out-of-band. Pár správ QUEUE_GET_CONFIG_REQUEST a QUEUE_GET_CONFIG_REPLY je mechanizmus, pomocou ktorého sa riadiací systém dozvie z prepínača, ako je nakonfigurovaná daná fronta. Na základe týchto informácií môže riadiaca jednotka inteligentne mapovať určité toky do špecifických frontov na dosiahnutie požadovaných úrovní QoS.

V prípade, ze HELLO zistí stratu spojenia medzi kontrolérom a prepínačom, špecifikácie vo verzií OpenFlow 1.0 predpisuje, že prepínač by mal vstúpiť do núdzového režimu a resetovať TCP spojenie. Všetky toky sa dajú v tomto okamihu vymazať, s výnimkou špeciálnych tokov, ktoré sú označené ako súčasť vyrovnávacej pamäte núdzového toku [4].

1.4.10 Programovanie tabuľky tokov

Obr. 1.4 zobrazuje ako kontrolér vykonáva dve jednoduché úpravy tabuľky tokov. Vidíme, že počiatočná tabuľka má tri toky. V kľudnom stave pre časom t_a vidíme rozložený tok 0. Ukazuje, že vstup toku špecifikuje, že všetky ethernetové rámce, ktoré prepínajú vstupný port K s cieľovou ethernetovou adresou, by mali byť výstupom na výstupnom porte N. Všetky ostatné polia zhody boli zastúpené znakmi, ktoré sú označené hviezdičkami v ich príslušných poliach zhody na Obr. 1.4. V čase t_a kontrolér vyšle do prepínača príkaz FLOW_MODE (ADD), ktorý pridá tok pre pakety vstupujúce do prepínača na ktoromkoľvek pote, so zdrojovými IP adresami 192.168.1.1 a cieľovou Ip adresou 209.1.2.1, zdrojovým TCP portom 20 a cieľovým portom 20. Všetky ostatné polia zhody sú zastúpené znakmi(hviezdičky). Post Outport je špecifikovaný ako P. Vidíme, že po prijatí a spracovaní tohto príkazu od

kontroléra na prepínač pribudne v tabuľke tokov nový vstupný tok F, ktorý koreš-ponduje zo správou ADD. V čase t_b ovládač vyšle príkaz FLOW_MOD pre nultý vstupný tok. Kontrolér sa snaží modifikovať vstup toku tak, že je tu hodinový (3600 sekúnd) čas nečinnosti. Po prepnutí tohto príkazu prepínačom bol pôvodný záznam toku upravený, tak aby odrážal tento nový čas nečinnosti. Čas nečinnosti pre vstup toku znamená, že po tomto časovom úseku by mal prepínač ukončiť tok dát na tomto toku. Správa FLOW_REMOVED naznačuje ze postup naprogramovaný v čase t_b uplynul. Tento kotrolér požaduje byť informovaný o tomto uplynutí platnosti, keď tok je nakonfigurovaný a správa FLOW_REMOVED slúžila tomuto účelu.



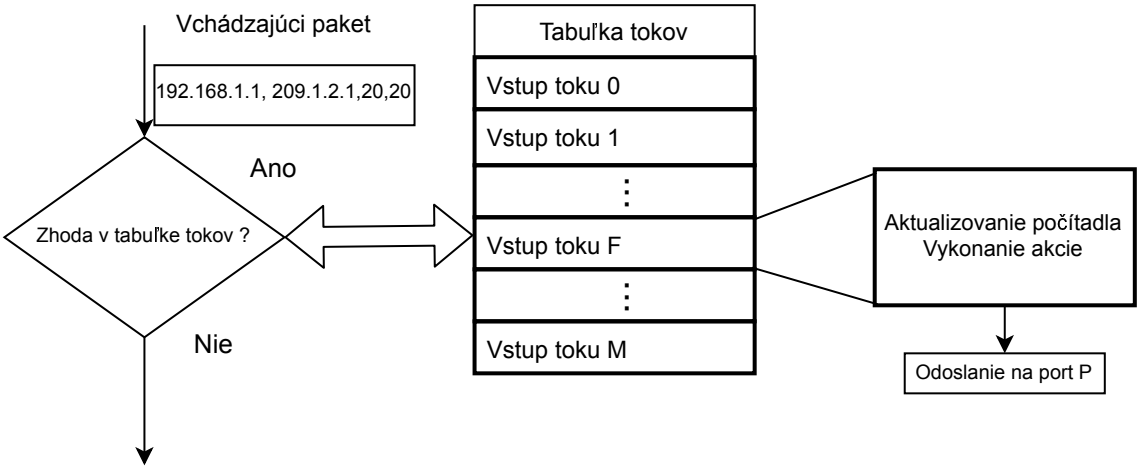
Obr. 1.4: Programovanie tabuľky tokov [20]

1.4.11 Základné posielanie paketov

Na Obr. 1.5 znázorňujeme najzákladnejší prípad posielania paketov. Paket zobrazený na Obr. 1.5 prichádza k prepínaču so zdrojovou IPv4 adresou 192.168.1.1 a koncovou IPv4 adresou 209.1.2.1. Funkcia porovnávania paketov prehľadáva tabuľku tokov

začínajúca vstupom toku 0 a nájde zhodu v vstupe toku F. V zázname toku F je uvedené, že zodpovedajúci paket by sa mal poslať ďalej portu P. Po dokončení tohto úkonu, prepínač dokončí základné posielanie paketov.

Prepínač sa rozhoduje na základe hlavičky. Programátor siete určuje správanie prepínača na tretej úrovni pomocou vstupov toku, a pokúša sa nájsť zhodu s hlavičkou paketu na tretej úrovni, ako napríklad IPv4. Ak je to prepínač druhej úrovne, vstupy toku sú určené na základe zhody na druhej úrovni. Schémy záznamu tokov umožňujú porovnávanie širokej škály hlavičiek protokolov, ale daný prepínač bude naprogramovaný iba pre tie, ktoré korešpondujú úlohe pri zasielaní paketov. Kedykoľvek dôjde k prekrývaniu potenciálnych zhôd tokov, priorita priradená vstupu toku priradená kontrolérom určí, ktorá zhoda ma prednosť. Napríklad, ak prepínač pracuje na druhej a tretej vrstve, nastavenie väčšej priority na tretiu úroveň zabezpečí, pokiaľ je možné, tak sa paket odošle na tejto úrovni.



Obr. 1.5: Posielanie paketov [21]

1.4.12 Zasielanie paketov do kontroléra

Ďalšia zásadná činnosť prepínača je zasielanie paketov do kontroléra na spracovanie výnimiek. Dva dôvody prečo môže prepínač poslať paket do kontroléra, sú OFPR_NO_MATCH a OFPR_ACTION. OFPR_NO_MATCH sa používa, keď sa nenájde žiadny záznam zodpovedajúceho toku. OpenFlow si zachováva schopnosť určiť, že konkrétny záznam toku by mal byť vždy zaslaný do kontroléra. V tomto prípade sa uvedie ako dôvod OFPR_ACTION. Príkladom by mohol byť riadiaci paket, ako je paket smerovacieho protokolu, ktorý musí vždy spracovať kontrolér. Na 5.10 je zobrazený príklad paketu prichádzajúcich údajov, ktorých zdrojom je smerovací protokol OSPF. Pre tento paket existuje zodpovedajúci záznam v tabuľke tokov, ktorý špecifikuje, že paket by sa mal poslať do kontroléra. Správa PACKET_IN je

zaslaná cez zabezpečený kanál do kontroléra, čo odovzdá túto aktualizáciu smerovaciemu protokolu kontroléru na spracovanie výnimiek. Proces,Fby ktorý sa vykoná na ovládači, by mal byť uskutočnený OSPF protokolom. Na tento paket existuje záznam zodpovedajúcej tabuľky, ktorý špecifikuje, že paket by sa mal poslať do kontroléra. Správa PACKET_IN ja zaslaná cez zabezpečený kanál do kontroléra, čo odovzdáva túto aktualizáciu smerovacieho protokolu kontroléru na spracovanie výnimiek. Spracovanie, ktoré by sa pravdepodobne uskutočnilo v kontroléri, by malo byť uskoutočnené smerovacím protokolom OSPF, čo by potenciálne mohlo viesť k zmene tabuliek posielania v prepínači. Kontrolér by potom mohol modifikovať tabuľky preposielania prostredníctvom príkazu FLOW_MOD vysielaného do prepínača modifikujúceho výstupný port pre každý port v prepínači ovplyvneným touto zmenou smerovacej tabuľky.

Kontrolér potrebuje minimálne jeden prístup do poľa záhlavia paketu, aby určil dispozíciu paketu. V mnohých prípadoch, aj ked nie vo všetkých, môže byť potrebný prístup k celému paketu. V záujme efektívnosti umožňuje OpenFlow voliteľné vyrovnávanie buffer pamäte celého paketu prepínačom. V prípade veľkého počtu paketov, ktoré sa posielajú z prepínača do kontroléra, u ktorého kontrolér potrebuje preskúmať len záhlavie paketu, sa dosiahne posielaním záhlavia paketu. Kontrolér niekedy bude musieť vidieť zostatok paketu, buffer ID komunikuje so správou PACKET_IN. Toto buffer IP môže kontrolér použiť na následné získanie celého paketu z prepínača[22].

1.5 SDN kontroléry

Kontroléry sú vybrané na základe dokumentácie a aktuálnosti. Sú to:

- ONOS
- OpenDayight
- Ryu
- Flood-Light
- Maestro

1.5.1 ONOS Open Network Operating System

ONOS bol navrhnutý tak, aby ponúkal flexibilitu pri vytváraní a nasadzovaní nových dynamických sieťových služieb zo zjednodušenými programovými rozhraniami. ONOS podporuje konfiguráciu aj riadenie siete v reálnom čase, čím eliminuje potrebu spúšťať protokoly riadenia smerovania a a prepínania v sieťovej štruktúre. Presunutím inteligencie do cloudového kontroléra ONOS je umožnená inovácia a

koncoví používatelia môžu ľahko vytvárať nové sieťové aplikácie bez potreby meniť dataplane systémy [23].

Platforma ONOS zahŕňa:

- Platforma a sada aplikácií, ktoré fungujú ako rozšíriteľný, modulárny, distribuovaný kontrolér SDN.
- Zjednodušená správa, konfigurácia a nasadenie nového softvéru, hardvéru.
- Škálovatelná architektúra, ktorá poskytuje odolnosť a škálovatelnosť požadovanú na splnenie prísnych podmienok v reálnom prostredí.

1.5.2 OpenDaylight

Hlavné rozdiely v OpenDaylight SDN v porovnaní s inými možnosťami SDN sú:

- Architektúra mikroprocesorov, v ktorej „mikroservis“ je konkrétny protokol alebo služba, ktorú chce používateľ povoliť v rámci svojej inštalácie OpenDaylight.
  - Doplnok, ktorý poskytuje pripojenie k zariadeniam prostredníctvom OpenFlow protokolov.
  - Platformová služba, ako napríklad overenie, autorizácia a účtovníctva.
  - Sieťová služba poskytujúca pripojenie VM pre OpenStack.
- Podpora širokého a rastúceho množstva sieťových protokolov: OpenFlow, BGP, Netconf, SNMP...
- MD-SAL (Model Driven Service Abstraction Layer). Yang modely zohrávajú v OpenDaylight kľúčovú úlohu a používajú sa pre:
  - Vytváranie datastore schémat.
  - Vytváranie aplikácií.
  - Automatické generovanie kódu.

OpenDaylight je napísaný v programovacom jazyku JAVA a je vhodný na cloudové riešenia[25].

1.5.3 Ryu

Ryu poskytuje softvérové komponenty s dobre definovanými API, ktoré vývojárom uľahčuje vytváranie nových aplikácií na správu a riadenie siete. Ryu podporuje rôzne protokoly pre správu sieťových zariadení, ako sú OpenFlow, Netconf.... Ryu podporuje OpenFlow verzie 1.0, 1.2, 1.3, 1.4, 1.5. Ryu je napísaný v programovacom jazyku Phyton[24].

1.5.4 Floodlight

Floodlight je open community kontrolér(je vyvíjaný otvorenou komunitou developerov) s apache licenciou(môže byť využitý takmer na akýkoľvek účel). Je navrhnutý na prácu s rastúcim počtom prepínačov, smerovačov, virtuálnych prepínačov a prístupových bodov, ktoré podporujú štandard OpenFlow. Floodlight je naprogramovaný v jazyku Java[27].

1.5.5 Maestro

Maestro je operačný systém na organizovanie aplikácií a riadenia siete. Poskytuje rozhrania na implementáciu aplikácií modulárneho riadenia siete na prístup a úpravu stavu siete a koordináciu ich interakcií. Maestro je platforma na dosiahnutie automatických a programových funkcií sieťového riadenia pomocou týchto modularizovaných aplikácií. Maestro sa neobmedzuje len na OpenFlow siete. Maestro poskytuje rozhranie pre:

- Zavádzanie nových prispôsobených ovládacích funkcií pridaním modularizovaných ovládacích komponentov.
- Zachovanie stavu siete v mene ovládacích komponentov.
- Zostavenie ovládacích komponentov zadaním post

Okrem toho sa Maestro snaží využiť paralelizmus v rámci jedného stroja na zlepšenie výkonu systému. Základnou črtou siete OpenFlow je to, že kontrolér je zodpovedný za počiatočné zriadenie každého toku kontaktovaním súvisiacich prepínačov. Výkon kontroléra by nemal byť prekážkou. Pri navrhovaní programu Maestro sa vyžaduje od programátorov čo najmenšie úsilie na zvládnutie parametrizácia. Namiesto toho Maestro rieši väčšinu zdĺhavej a komplikovanej úlohy riadenia distribúcie pracovnej záťaže a plánovania pracovných vlákien[26].

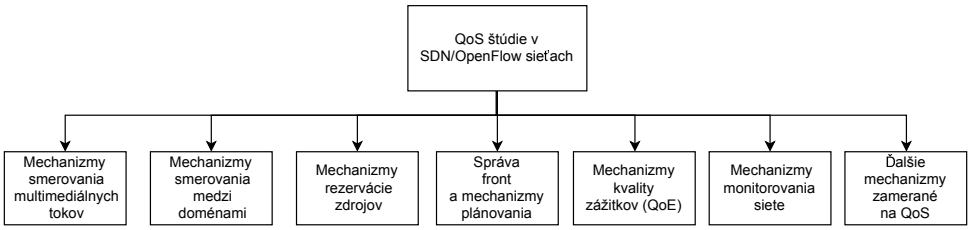
1.6 QoS v SDN sieti

V prípade niektorých sietí, najmä v prípadoch vojenských služieb a služieb prvej relácie, je dôležite kontrolovať, kto získa prístup k sieťovým zdrojom a a zabezpečiť správne služby vybraným používateľom, je tiež dôležité mať prísne oddelenie používateľov, aby sekundárni používatelia neporušovali služby pre primárnych používateľov.

SDN ponúka nové možnosti na splnenie týchto úloh. Najvýznamnejšou črtou SDN je oddelenie riadiacej a dátovej roviny. Riadiaca rovina môže poskytovať centralizovaný pohľad na sieť, ktorá je riadená prostredníctvom southbound rozhrania SDN kontroléra. Dátová rovina preposiela alebo zahadzuje pakety na základe

pravidiel toku a klasifikátorov určených riadiacou rovinou. Pretože SDN kontrolér môže potenciálne získať globálny pohľad na sieť a stav, v prípade potreby je možne špecifikovať pravidlá riadenia a dohody o úrovni služieb (SLA). Tieto mechanizmy prinášajú nové príležitosti pre niekoľko funkcii QoS, ako je QoS riadenie rámcov, monitorovanie... Jeden z priestorov uplatnenia QoS je uprednostnenie jedného konkrétneho typu prenosu pred ostatnými druhmi prenosu. Aby sa táto úloha mohla realizovať, sieť musí poskytovať prostriedky na udelenie jednej sady prístupu používateľa k niektorým privilegovaným zdrojom, zatiaľ čo ostatným sa bráni v prístupe k týmto zdrojom. Na dosiahnutie tohto cieľa musia byť klienti autentifikovaní na základe svojich poverení a musia byť autorizovaní do konkrétnej virtuálnej siete a oddelení od ostatných používateľov. Hoci SDN poskytuje niekoľko stavebných blokov na splnenie týchto úloh, nie je jednoduché poskytnúť integrované riešenie, ktoré kombinuje riadenie prístupu, oddelenie a prioritizáciu QoS [7].

Na Obr. 1.6 je znázornené ako sú organizované súvisiace štúdie do siedmich kategórií, v ktorých môže QoS ťažiť z SDN koncepcie. Tieto štúdie sú: mechanizmy smerovania multimediálnych tokov, mechanizmy smerovania medzi doménami, mechanizmy rezervácie zdrojov, správa front a mechanizmy plánovania, mechanizmy kvality zážitkov (QoE), mechanizmy monitorovania siete a ďalšie mechanizmy zamerané na QoS, ako je zabezpečenie QoS založené na na vitualizácii a riadenie politiky QoS atď.



Obr. 1.6: QoS [8]

Prvé dva typy mechanizmov sú poháňané funkciou smerovania. Tretí a štvrtý typ mechanizmov sa sústreďuje okolo rezervovania zdrojov a riadenia front a plánovania paketov na podporu QoS. Piaty typ štúdie sa zameriava na QoE systém, zatiaľ čo šiesta skupina štúdií sa točí okolo rámcov monitorovania siete. Posledná skupina študuje rôzne problémy súvisiace s QoS, ako je riadenie politiky QoS, rozšírenia testovacích QoS atď[8].

1.6.1 Vzťah medzi QoS a SDN

QoS je zvyčajne definované ako schopnosť siete poskytovať služby pre vybranú sieťovú prevádzku. Hlavným cieľom QoS je poskytnúť prioritu na parametre QoS, ale

bez obmedzenia:

- šírka pásma
- oneskorenie
- jitter
- straty

Na zabezpečenie QoS je potrebné rozlišovať aplikačné toky, pretože bojujú o dostupné sieťové zdroje. Tieto sieťové zdroje sa musia prideliť, aby sa zabezpečila priorita prenosu s vyššou prioritou pre vhodné rozdelenie sieťových prostriedkov. Tento proces často vyžaduje znalosť súčasných stavov siete, aby bolo možné robiť správne rozhodnutia, pokiaľ ide o zasielanie paketov.

V súčastnosti sa poskytovanie QoS väčšinou spolieha na dohody o úrovni služieb (SLA) medzi koncovými používateľmi a poskytovateľmi služieb. Tento prístup funguje dobre pre best–effort služby a nepodporuje jemnejšie riadenie toku. Existujú však aj iné typy aplikácií, ako napríklad VoIP, hranie online hier a videokonferencie, ktorých toky sú citlivé na oneskorenie, jitter a šírku pásma, a preto si vyžadujú špeciálne zaobchádzanie. Aj architektúra rozhodovania typu „hop-by-hop“ sa niekedy ťažko monitoruje, najmä kvôli použitiu mnohých rôznych firmvérových značiek. Neexistuje štandardizovaný spôsob, ako špecifikovať politiku a obmedzenia riadenia toku na vysokej úrovni, pokiaľ ide o hĺbkové odlíšenie toku dát.

QoS je implementované hlavne v dvoch prístupoch: hard QoS a soft QoS. Metóda hard QoS zaručuje požiadavky QoS na pripojenie, ale trpí obmedzeniami zdrojov. IntServ metóda je príkladom poho typu prístupu QoS zaručujúceho prístup. Na druhej strane metóda soft QoS nieje taká prísna ako metóda hard QoS, pokiaľ ide o požiadavky na QoS. DiffServ je príkladom metódy soft QoS[8].

1.7 Netconf

Network configuration protocol (Netconf) je API southbound rozhranie pre SDN aplikácie. Netconf bol vyvinutý za účelom nahradenia SNMP protokolu. Jazyk XML (Extensible Markup Language) sa používa na odovzdávanie príkazov a dát do a zo sieťových zariadení. Existuje niekoľko silných stránok, pre ktoré je Netconf vhodný pre správu siete aj pre SDN:

- Zabezpečenie: Netconf od začiatku fungoval prostredníctvom zabezpečeného kanála.
- Organizácia: Netconf oddeľuje konfiguračné a prevádzkové údaje, vďaka čomu je organizácia a kontrola rôznych častí údajov oveľa jednoduchšia.
- Oznámenie schopností: Prvá výmena medzi serverom Netconf na zariadení a klientom na riadiacej stanici je oboznámenie o všetkých Yang modeloch, ktoré

zariadenie podporuje. Neexistuje teda nejednoznačnosť, pokiaľ ide o to, ktoré funkcie zariadenie podporuje.

- Operácie: Operácie Netconf sú vzdialené volania procedúr (Remote Procedure Calls-RPC). Táto schopnosť umožňuje SDN kotroléru, aby dal zariadeniu pokyn, na prijatie konkrétnej akcie, a odovzdalo sadu parametrov do RPC. Toto umožňuje kontroléru ľahšie manipulovať s preposielaním správania na zariadenie.

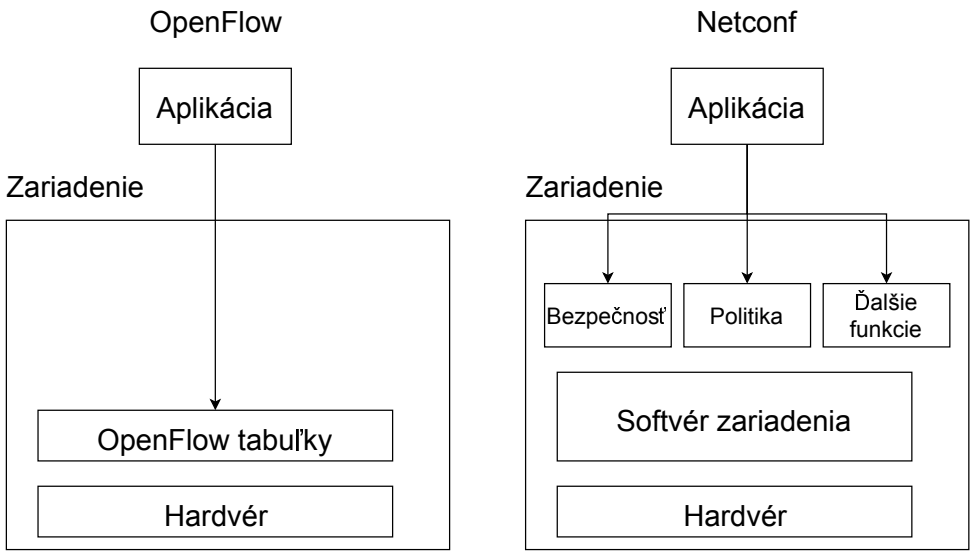
SDN API ako Netconf sa teší veľkej obľube medzi veľkými poskytovateľmi služieb, čo viedlo k tomu aby dodávatelia ako Jumper a Cisco implementovali zariadenia Netconf zamerané na týchto zákazníkov. Toto vytvorilo základ pre podporu Netconf v týchto prostrediach.

Jedno z nevýhod je, že Netconft trpí nedostatkom štandardných modelov Yang. Napríklad SNMP mal od začitku definíciu MIB pre veľa kritických častí údajov na prepínačoch a smerovačoch. V porovnaní s tým ze priemysel sa stále snaží vytvoriť štandardné modely Yang pre smerovanie, prepínanie, rozhodovanie, ACL a ďalšie základné sieťové komponenty. To bráni vývojárom SDN, ktorí musia písať kód špecificky pre dané zariadenie, aby mohli ovládať rôzne modeli Yang, ktoré existujú pre rovnakú funkcionalitu. Napriek tomu je dokázané, že Netconf bude jedným z primárnych protokolov požívaných pri vývoji SDN aplikácií[28].

1.7.1 Použitie Netconf u SDN

Protokol bol navrhnutý ako nástupca protokolu SNMP a pokúsil sa vyriešieť niektoré nedostatky protokolu SNMP. Kľúčové atribúty Netconf sú:

- Oddelenie konfiguračných a stavových údajov. Konfiguračné údaje sú nastavené na zariadení, aby fungovali určitým spôsobom. Stavové (prevádzkové) údaje nastavuje zariadenie ako výsledok dynamických zmien na zariadení v dôsledku udalostí a aktivít v sieti.
- Podpora pre funkcie podobné vzdialenému volaniu procedúr (RPC). Takáto funkcia nebola v SNMP k dispozícii. S Netconf je možné vyvolať operáciu na zariadení, odovzadnie parametrov a prijímanie vrátených výledkov, podobné ako volania RPC v programovacej paradigme(súbor princípov, na ktorých je založený programovací jazyk).
- odpora upozornení. Táto schopnosť je všeobecným mechanizmom udalostí, pomocou ktorého môže riadený prístroj upozorniť riadiacu stanicu na významne udalosti. V rámci SNMP sa tento koncept nazýva pasca(trap).
- Podpora konfigurácií založených na transakciách. Toto umožňuje, aby sa začala konfigurácia násobných zariadení, ale potom sa vrátila späť v prípade zlyhania v určitom okamihu procesu.



Obr. 1.7: OpenFlow vs Netconf

Netconf je protokol riadenia a ako taký má schopnosť konfigurovať iba tie možnosti ktoré sú zariadeniam vystavené. Na Obr. 1.7je zobrazený rozdiel medzi zariadením riadeným Netconf a zariadením riadeným OpenFlow. OpenFlow konfiguruje nižšie úrovne sieťového zariadenia, to je ASIC, ktorý obsahuje TCAM. To znamená nastavenie zápisov a akcií FIB.

Na druhej strane Netconf vykonáva tradičnú konfiguráciu zariadenia, ale prostredníctvom Netconf protokolu namiesto CLI alebo SNMP. Vývojár aplikácie SDN je obmedzený tým, čo zariadenie vystavuje ako konfigurovateľné. Ak zariadenie odhaľuje Netconf dátové modely, ktoré umožňujú konfiguráciu prístupových zoznamov (ACL), potom ich aplikácia môže nakonfigurovať. Podobne, ak ma zariadenie dátové modely na konfiguráciu QoS alebo statickej trasy, potom budú možne aj tieto.

Informácie o funkciách je možné získať prostedníctvom dátových modelov Yang, ktoré zariadenie podporuje[29].

1.7.2 Yang

Samotný Netconf je protokol na správu siete, rovnako ako SNMP. Niektoré protokol sa stávajú užitočnými prostrdníctvomm dátových modelov, ktoré prinášajú informácie a požiadavky až do zariadenia. V prípade SNMP mali dátové modely formu riadiacej informačnej základne (MIB), ktorú sme definovali pomocou štrukúry informácií o riadení (SMI). V porovnaní Netconf a SNMP možeme povedať ze Yang je analógia na SMI a Yang model je anlógia na MIB.

Yang poskytuje štandardizovaný spôsob, ako zariadenia podporujú a propagujú svoje schopnosti. Jedno z prvých operácií, ktorá prebieha medzi klientom Netconf

na kontroléri a Netconf Serverom bežiacom na zariadení, je to, aby zaridenie informovalo klienta o tom, ktoré dátové modely sú podporované. To umožňuje SDN aplikácií bežiacej na kontroléri zistiť, ktoré operácie sú možné na jednotlivých zariadeniach. Táto informácie je kľúčová, pretože rôzne zariadenia sa často budú líšiť vo svojich schopnostiach. Jednou zo súčasných nevýhod Netconf sietí a Yang je to , že rôzni dodávatelia často podporujú odlišné Yang modely. Toto sa môže vyskytnú aj v rámci rôznych skupín výrobkov od toho istého výrobcu. Na rozdiel od SNMP a štandardom MIB v súčasnosti neexistuje konzistentný sada dátových modelov Yang podrobných naprieč priemyslom. V súčastnosti je potrebné, aby aplikácie požadovali a nastavovali údaje na rôznych zariadeniach.

Modely Yang sú stále relatívne nové a je pravdepodobné, že štandardizované modely budú definované v blízkej budúcnosti. Uľahčí to skutočnosť, že moderné sieťové majú lepšie interné konfiguračné schémy ako v prvých dňoch SNMP, takže pre väčšinu výrobcov je relatívne ľahké generovať modely Yang, ktoré mapujú tieto schémy[30].

1.8 NFV: Virtualizácia sieťových funkcií

Je pravdepodobné, že vitualizácia sieťových funkcií (NFV) stavia na najdôležitejších koncepciách SDN. Medzi ne patrí oddelenie kontrólnej/dátovej roviny, logická centralizácia, kontrolér, sieťová virtualizácia (logické prekrytia), informovanosť o aplikácií, kontrola zámeru aplikácie a trend ich spúšťania na komoditných hardvérových platformách. NFV rozširuje tieto koncepcie o nové metódy na podporu prepojenia prvkov služieb, ako je napríklad reťazec servisných funkcií (SFC), a nové techniky riadenia, ktoré sa musia použiť na zvládnutie dynamických a pružných schopností.

V dnešnej sieťovej infraštruktúre existujú rôzne typy proprietárnych hardvérových zariadení na vykonávanie základných sieťových operácií, Pridanie novej služby do takejto architektúry je dosť ťažké, pretože si vyžaduje umiestnenie zariadenia, ktoré túto službu hosťuje, vzájomné fungovanie zariadení a niekedy prepracovanie celej architektúry. Navyše používanie vyhradených špeciálnych zariadení pre každú funkciu nie je nákladovo efektívne[9].

NFV je je nová technológia, ktorá poskytuje riešenia uvedených problémov oddelením sieťových funkcií od vyhradeného hardvéru a ich implementáciu ako softvéru na veľkých štandardných serveroch. S NFV sú sieťové funkcie virtualizované a ľahko implementované na vhodných miestach bez použitia špeciálneho hardvérového zariadenia pre každú službu. NFV odstraňuje závislosť na hardvéri ako v SDN a znižuje náklady na hardvér a spotrebu energie. Inovácie sa zrýchľujú a nové služby sa na trh zavádzajú v kratšom čase, pretože sa poskytujú ako softvér. Ďalej je možné

medzi službami zdielať zdroje, znižuje sa potreba pracovnej sily a služby sa môžu premiestňovať v sieti bez potreby inštalácie hardvéru[10].

1.8.1 Vztah medzi SDN a NFV

Hoci SDN a NFV môžu byť implementované nezávisle jeden od druhého, sú doplnkovými a podpornými technológiami. Majú veľa spoločných cieľov, pokiaľ ide o softvér, programovateľnú sieťovú infraštruktúru a ľahšiu prevádzku a údržbu.

SDN centralizuje rozhodvací proces v sieti využitím výhod programovateľnosti zariadení na posielanie dát, zatiaľ čo NFV virtalizuje sieťové funkcie a implementuje ich ako softvér na univerzálne vysokokapacitné servery. Inými slovami, SDN poskytuje programovateľnú sieť, kde je možné efektívne využívať VNF(Virtual Network Function). Spoločne umožňujú SDN a NFV flexibilné prostredie, v ktorom VNF a prevádzkové požiadavky ovplyvňujú pravidlá smerovania toku dát. Hlavnou črtou SDN používanou na umiestnenie VNF je jej schopnosť rýchlo a dynamicky inštalovať konkrétne trasy, aby sa umožnilo reťazenie servisných funkcií. Nasadenie funkčných reťazcov služieb a smerovanie dátových tokov by bolo v SDN oveľa jednoduchšie vďaka rýchlejšej aktualizácií a ľahšej údržbe, ktorú poskytuje programovateľná sieťová infraštruktúra pod centrálnou kontrolou. Aby to bolo možné podrobnejšie vysvetliť, dopravné toky je možné prideliť reťazcom servisných funkcií a požadované pravidla posielania môžu byť rýchlo zapísané do tabuľky tokov všetkých programovateľných zariadení pomocou centralizovaného SDN kontroléra, aby sa zabezpečilo, že každý tok sleduje správny reťazec servisných funkcií. Z týchto dôvodov je SDN akceptovaná ako ideálna platforma pre NFV a sieťové architektúry, ko sú architektúry navrhované pre 5G, ktoré predpokladajú koexistenciu technológií SDN a NFV[10].

1.9 Virtualizácia v oblasti dátových centier

Počítačová virtualizácia sa veľmi nevyužívala, pokiaľ nezačali prevažovať dátové centrá a potreba dynamicky vytvárať a rušiť servery, presúvať ich z jedného fyzického servera na iný. Po tomto sa práca dátového centra od základu zmenila. Servery sa spúšťali jedným kliknutím myši a mohli sa presúvať bez toho aby sa narušila činnosť premiestňovaného servera. Vytvorenie nového virtualného počítača alebo presunutie virtualného počítača z jedného fyzického servera na druhý, je z pohľadu správcu servera jednoduché, a je ho možné vykonať veľmi rýchlo.

Virtualizačné softvéri ako napríklad VMware, Hyper-V, KVM a XenServer sú príkladom produktov, ktoré umožňujú správcovi servera ľahko vytvárať alebo presúvať virtualne stroje. Týmto sa skrátil čas, potrebný na spustenie novej inštalácie

servera na niekoľko minút, či dokonca sekúnd.

Virtualizácia úložného priestoru existuje už dosť dlhú dobu, rovnako aj ako koncepcia oddeľovania blokov ukladania a umožnenie ich oddelenia od skutočného hardvéru fyzického ukladacieho priestoru. Podobne ako u serverov sa tu dosahuje účinnosť z hľadiska rýchlosti(presun často používaných údajov na rýchlejšie zariadenie) ako aj z hľadiska využívania (umožnenie viacerým serverom zdielať rovnaké úložné zariadenie). Tieto technologické vylepšenia umožňujú rýchlu a efektívnu manipuláciu so servermi a úložiskami [31].

2 Výsledky študentskej práce

2.1 OpenDaylight

Ako bolo spomenuté v časti o SDN kontroléroch najvhodnejším kontrolérom pre riešenie zadania je OpenDaylight. Tento kontrolér je písaný v jazyku JAVA a je vhodný pre cloudové riešenia[25].

2.1.1 Koncepty a nástroje OpenDaylight

Základné koncepty a nástroje:

- Vývojári OpenDaylight doposiaľ vytvorili viac ako 50 projektov zameraných na rozšírenie spôsobu fungovania siete. Projekty sú formálnou štruktúrou, na ktorej sa vývojári stretávajú, dokumentujú plány uvolnenia kódu a uvoľňujú funkcie, ktoré vytvárajú v OpenDaylight.
- **Apache Karaf** poskytuje ľahký runtime na inštaláciu funkcií Karaf, ktoré chceme implementovať, a je súčasťou softvérovej platformy OpenDaylight. V predvolenom nastavení nemá OpenDaylight predinštalované funkcie.
- Model-Driven Service Abstraction Layer (MD-SAL) je rámec OpenDaylight, ktorý umožňuje vývojárom vytvárať nové funkcie Karaf vo forme ovládačov služieb a protokolov a vzájomne ich spájať. MD-SAL obsahuje tieto dva komponenty:
  - Zdieľané dátové úložisko, ktoré spravuje nasledujúce stromové štruktúry:
    - Config Datastore, ktorý udržuje reprezentáciu požadovaného stavu siete.
    - Prevádzkové dátové úložisko, ktoré predstavuje aktuálny stav siete na základe údajov z riadených prvkov siete.
  - Zbernica správ, ktorá poskytuje rôzne ovládače služieb a protokolov na vzájomné oznámenie a komunikáciu.
- Komunikácia s OpenDaylight prostredníctvom rozhrania REST API a počas používania OpenDaylight rozhrania, architektúra mikrosevices umožňuje vybrať dostupné služby, protokol a REST API.

2.1.2 Clustering

Clustering je mechanizmus, ktorý umožňuje viacerým procesorom a programom spolupracovať ako jedna entita. Napríklad, vyhľadávanie na google.com, môže vyzerať, že žiadosť na vyhľadanie spracuje len jeden webový sever. V skutočnosti je požiadavku na vyhľadávanie spracovávajú ľubovolné webové servery prepojené clustermi.

Podobne môže mať niekoľko inštalácií aj OpenDaylight pracujúci spolu ako jedna entita.

Výhody clusteringu:

- Škálovanie: Ak je spustených viacero inštalácií OpenDaylight, je možne potenciálne robiť viac práce a ukladať viac údajov, ako by sa dokázalo s jednou inštaláciou. Je možne tiež rozdeliť na menšie kúsky a tieto potom distribuovať do clustera alebo vykonať určité operácie s určitými členmi clustera.
- Vysoká dostupnosť: Ak je spustených viacero inštalácií OpenDaylight a jedna z nich zlyhala, stále budú k dispozícií ďalšie pracujúce inštalácie.
- Perzistencia údajov: po manuálnom reštarte alebo havárií sa nestratia žiadne údaje uložené v OpenDaylight.

2.1.3 Bezpečnosť

Bezpečnostné vysokoúrovňové výhody OpenDaylight:

- Oddelenie kontrolných a riadiacich rovín do dátovej roviny umožňuje vynútenie možných bezpečnostných hrozieb na menšiu plochu útoku.
- Centralizované riadenie informácií a sietí poskytuje správcom siete väčšiu viditeľnosť a kontrolu nad celou sieťou, čo im umožňuje lepšie a rýchlejšie rozhodnutia. Centralizácia sieťového riadenia môže byť zároveň výhodou iba vtedy, ak je zabezpečený prístup k tomuto ovládaniu.
- Schopnosť rýchlejšie rozvíjať southbound protokoly a spôsob ich používania poskytuje stále rýchlejšie mechanizmy na uzákonenie vhodných opatrení na zmiernenie a nápravu bezpečnosti.
- OpenDaylight je postavený a OSGi zväzkoch a Karf Java kontajneri. Karf aj OSGi poskytujú určitú úroveň izolácie s explicitnými hranicami kódu, importom paketov, exportom paketov, a ďalšími bezpečnostnými prvkami
- OpenDaylight má históriu rýchleho riešenia známych slabých miest a dobre definovaného procesu ich hlásenia a riešenia.

2.2 Open vSwitch

V prípade hypervizorov založených na Linuxe sa na premostenie komunikácie medzi VM a okolitým svetom môže použiť rýchli a spoľahlivý L2 prepínač (Linux bridge). Toto riešenie nieje zlé ale v prípade použitia v multi-serverovej virtuálizácií nieje vhodný. Tieto prostredia sa často vyznačujú vysoko dynamickými koncovými bodmi, udržiavaním logických abstrakcií a v niektorých prípadoch integráciu s prepínacím hardware. V tomto prípade je vhodnejšie použiť OVS.

2.2.1 Mobilita

Celý stav siete spolu so všetkými sieťovými entitami (VM) by mal byť ľahko identifikovateľný a migrovateľný medzi rôznymi hostiteľmi. To môže zahŕňať tradičný „mäkký stav“ čo môžu byť záznamy v L2 tabuľke, stav smerovacej politiky, ACL, politiku QoS a podobne. OVS má podporuje konfiguráciu a migráciu pomalého (konfiguračného) a rýchleho stavu siete medzi hostiteľmi. Napríklad, ak VM migruje medzi koncovými hostiteľmi, je možne migrovať pridruženú konfiguráciu ale aj akýkoľvek stav živej siete. Stav OVS je typizovaný a podporovaný skutočným dátovým modelom, ktorý umožňuje vývoj štrukturovaných automatizovaných systémov.

2.2.2 Reakcia na dynamiku siete

Virtuálne prostredia sa vyznačujú vysokou mierou zmien. VM pribúdajú a ubúdajú, pohybujú sa v čase, menia sa logické sieťové rozhrania a tak ďalej. OVS podporuje množstvo funkcií, ktoré umožňujú systému riadenia siete reagovať a prispôsobiť sa pri zmene prostredia. Užitočnou výhodou je podpora OVSDB databázy stavu siete, ktorá podporuje vzdialené spúšťače. Pre toto môže riadiaci softvér sledovať a reagovať na rôzne aspekty siete, ak sa zemina. OVS tiež podporuje OpenFlow ako metódu exportovania vzdialeného prístupu na riadenie prenosu. Na tento účel existuje mnoho použití, vrátane globálneho zisťovania siete prostredníctvom kontroly objavovania alebo sledovania prenosu stavu spojenia.

2.2.3 Podporované platformy

OVS operuje na viacerých vizualizačných platformách a prepínacích chipseteoch. Je použitý ako predvolený prepínač v XenServer 6.0, na platforme Xen Cloud Platform a tiež podporuje Xen, KVM, Proxmox VE a VirtualBox. Je tiež integrovaný do mnohých vizualizačných riadiacich systémov ako napríklad OpenStack, openQRM, OpenNebila a oVirt. Jadro je distribuované v systéme linux a je dostupné pre Ubuntu, Debian, Fedora a openSUSE [32].

2.3 VMware ESXi

VMware ESXi server je vizualizačná platforma, v ktorej je možné vytvárať a prevádzkovať virtuálne stroje a virtuálne zariadenia. Je to jedna z dvoch komponent vSphere. Druhá komponenta je vCenter Server.

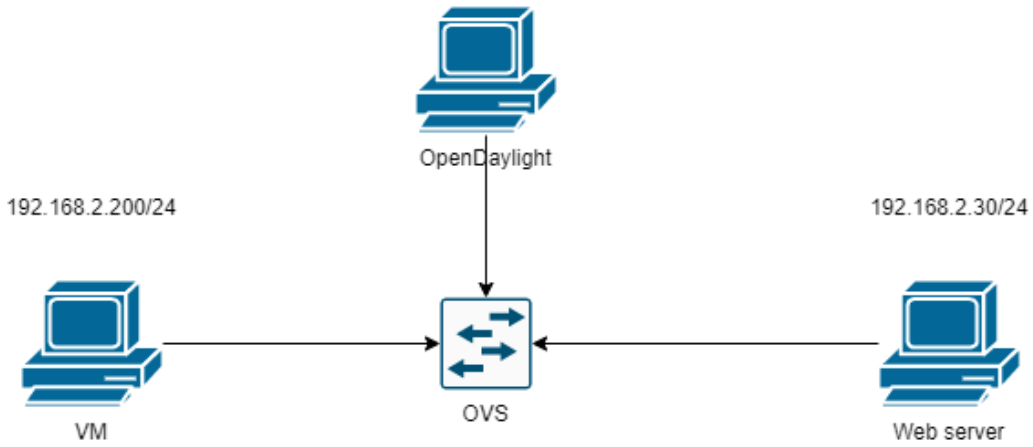
2.3.1 vSphere

VMware vSphere je vizualizačná platforma VMware, ktorá transformuje dátové centrá do agregovaných počítačových infraštruktúr, ktoré zahŕňajú prostriedky CPU, úložného priestoru a sieťových zdrojov. vSphere riadi tieto infraštruktúry ako jednotné operačné prostredie a poskytuje nástroje na správu dátových centier, ktoré sa zúčastňujú na tomto prostredí [33].

2.4 Topológia siete

Na realizáciu topológie by sa dal využiť Mininet, ktorý poskytuje prostredie na vytváranie sieťových topológii. V mojom prípade som išiel radšej reálnou cestou a vytvoril som virtuálne prostredie na ESXi servery. V tomto virtuálnom prostredí sa nachádzajú virtuálne počítače, ktoré slúžia na ovládanie, prepojenie alebo testovanie pripojenia v sieti. Na ovládanie siete slúži virtuálny počítač s názvom OpenDaylight, na prepojenie slúži virtuálny počítač s názvom OVS a na testovanie pripojenia sú dva virtuálne počítače s názvom VM a Web server. Topológia siete je zobrazená na obr. 2.1.

Cieľom tejto topológie je oboznámiť študenta, ktorý bude vypracovávať laboratórnu úlohu s vytáraním a modifikovaním tokov v sieti. Študent bude mať za úlohu vytvoriť toky na povolenie ICMP správ, toky na povolenie služieb ako sú SSH a HTTP a v posledom rade bude mať za úlohu modifikovať ICMP toky na povolenie pingu len z jedného počítača.



Obr. 2.1: Topológia laboratórnej úlohy.

2.5 Hardvérové riešenie

Vzhľadom na moje zadanie čo je „SDN pre cloud computing“, tak si táto úloha nevyžaduje veľké množstvo hardvérových komponentov. Na realizáciu je potrebný server, ktorý zabezpečí virtuálizáciu ako napríklad VMWare ESXi. Na tomto servery potom budú inštalované jednotlivé virtuálne počítače ako je spomenuté v Topológii siete.

2.6 Príprava prostredia

V tejto časti je popísane aké zariadenia a programy som použil na realizáciu topológie. Ako už bolo spomenuté na väčšinu riešenia nieje potrebný špeciálny hardware, preto tu je popísané len softvérové riešenie.

2.6.1 Ubuntu 18.04

Počítače s grafickým rozhraním sú realizované operačným systémom Linux Ubuntu 18.04 v desktop verzií v základnej inštalácií. Hlavným dôvodom pre výber operačného systému Ubuntu 18.04 je jeho prívetivé grafické rozhranie, v ktorom budú študenti pracovať, a skutočnosť ze inštalácia kontroléru je možná na systéme Linux a menšia náročnosť systému na hardware. V systéme je nastavené heslo pre užívateľa root student a pre užívateľa student, student. Toto nastavenie je rovnaké pre každý použitý systém v topológii.

2.6.2 Inštalácia OpenDaylight

Kontróler OpenDaylight je nainštalovaný na VM s názvom OpenDaylight. Operačný systém, na ktorom je OpenDaylight nainštalovaný je už spomínaný Ubuntu 18.04.

Inštalácia OpenDaylight je veľmi jednoduchá a dá sa zhrnúť do pár krokov:

- Pred samotnou inštaláciou som stiahol a nainštaloval Java a spravil export JAVA_HOME. Java som stiahol a nainštaloval pomocou príkazu :

  ```
  $ sudo apt-get -y install openjdk-8-jre
  ```

  Po nainštalovaní som zistiť, kde sa Java uložila. Toto som zistil pomocou zadania príkazu:

  ```
  $ sudo update-alternatives --config java
  ```

  Po zadaní príkazu sa zobrazí hláška:
  There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java. Na základe tohto výpisu som

vedel kde je Java uložená. Táto informácia je potrebná na správny export JAVA_HOME. Export JAVA_HOME som realizoval zadaním príkazu:

```
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
```

Po správnej inštalácií Java som nainštaloval kontrolér OpenDayLight.

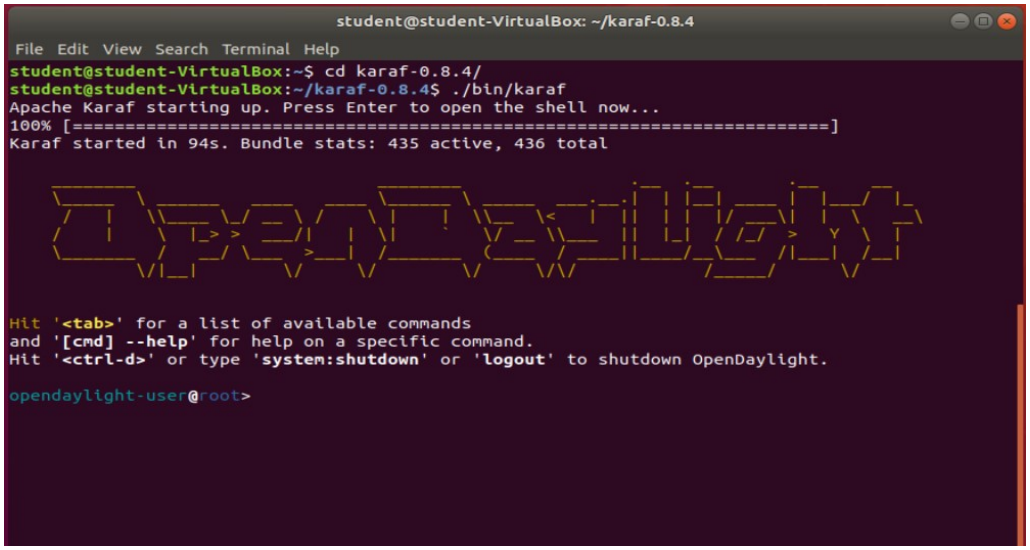
- Na stránke *https://docs.opendaylight.org/en/latest/downloads.html* sa nachádzajú inštalačné súbory pre najnovšie ale aj staršie verzie OpenDaylight. Ako prvé sú zobrazované najnovšie verzie OpenDaylight. Staršie verzie sa nachádzajú pod položkou *Archived Releases*. V tejto práci je použitý kontrolér OpenDaylight vo verzií 0.8.4 Oxygen. Zvolená je staršie verzia kvôli podpore grafického rozhrania DLUX. Novšie verzie už toto rozhranie nepodporujú. Po zadaní príkazu:

  ```
  $ wget https://nexus.opendaylight.org/content/
  repositories/opendaylight.release/org/
  opendaylight/integration/karaf/0.8.4/
  ```

  sa stiahne ZIP súbor ktorý obsahuje súbory na spustenie OpenDaylight. Stiahnutý ZIP som rozbalil pomocou príkazu:

  ```
  $ sudo unzip karaf-0.8.4.zip
  ```

  Po rozbalení súboru vznikne adresár s pomenovaním *karaf-0.8.4*. OpenDaylight som spustil pomocou príkazov na obr. 2.2.



Obr. 2.2: Spustenie OpenDaylight

2.6.3 Inštalácia OVS (Open Vswitch)

OVS je virtuálny prepínač, ktorý podporuje OpenFlow protokol. Inštalácia OVS je ešte jednoduchšia ako inštalácia OpenDaylight. OVS nepotrebuje odinštalovanie iných programov alebo rozšírení. Všetko prebehne automatiky po zadaní príkazu:

```
$ sudo apt install openvswitch-switch
```

Po inštalácii sa OVS spustil sám.
Spustenie som overil pomocou príkazu:

```
$ sudo ovs-vsctl show
```

Výpis z tohto príkazu zobrazí nainštalovanú verziu OVS a prípadne nastavenie pokiaľ je OVS nastavený.

2.6.4 Inštalácia OpenDaylight features

Po stiahnutí OpenDaylight niesu aktívne všetky funkcie, ktoré tento kontrolér ponúka. OpenDaylight ponúka veľké množstvo funkcií, z tohto dôvodu sa potrebné funkcie musia nainštalovať. Práca s kontrolérom je veľmi jednoduchá. Je možnosť využiť nápovedu pomocou klávesy TAB. Pre túto prácu sú potrebné fatures DLUX (grafické rozhranie), Openflow, l2switch, restconf, yangtools a MD-SAL. Na inštaláciu jednej funkcie je potrebné stiahnuť viacej feartures a následná inštalácia je realizovaná pomocou príkazu:

```
> feature:install Názov_feature
```

Feature DLUX zabezpečuje grafické rozhranie.V mojej práci sú potrebné len tieto features [34]:

```
odl-dlux-core
odl-dluxapps-topology
```

Feature OpenFlow slúži na komunikáciu s OpenFlow prepínačom OVS.V mojej práci som stiahol tieto features [35]:

```
odl-openflowjava-protocol
odl-openflowplugin-drop-test
odl-openflowplugin-libraries
odl-openflowplugin-nsf-model
odl-openflowplugin-southbound
odl-openflowplugin-app-topology
odl-openflowplugin-flow-services
odl-openflowplugin-nxm-extensions
odl-openflowplugin-onf-extensions
```

```
odl-openflowplugin-app-bulk-o-matic
odl-openflowplugin-app-lldp-speaker
odl-openflowplugin-app-config-pusher
odl-openflowplugin-app-notifications
odl-openflowplugin-app-southbound-cli
odl-openflowplugin-flow-services-rest
odl-openflowplugin-app-topology-manager
odl-openflowplugin-app-table-miss-enforcer
odl-openflowplugin-app-forwardingrules-sync
odl-openflowplugin-app-forwardingrules-manager
odl-openflowplugin-app-topology-lldp-discovery
odl-openflowplugin-app-arbitratorreconciliation
odl-openflowplugin-app-reconciliation-framework
```

Feature l2switch slúži na nastavenie OpenFlow prepínača OVS aby sa správal ako L2 prepínač.V mojej práci som nainštaloval túto feature [36]:

```
odl-l2switch-all
```

Feature restconf na vytvorenie komunikácie REST API ako napríklad OpenDaylight OpenFlow Manager. V mojej práci som nainštaloval túto feature:

```
odl-restconf-all
```

Feature yangtools slúži na podporu YANG v Java prostredí. V mojej práci som stiahol tieto features [37]:

```
odl-yangtools-data
odl-yangtools-util
odl-yangtools-codec
odl-yangtools-common
odl-yangtools-export
odl-yangtools-parser
odl-yangtools-data-api
odl-yangtools-parser-api
```

Feature MD-SAL slúži na odosielanie správ a ukladanie údajov na základe modelových údajov na rozhraní, ktoré je definované vývojármi aplikácie. V mojej práci som nainštaloval túto feature [38]:

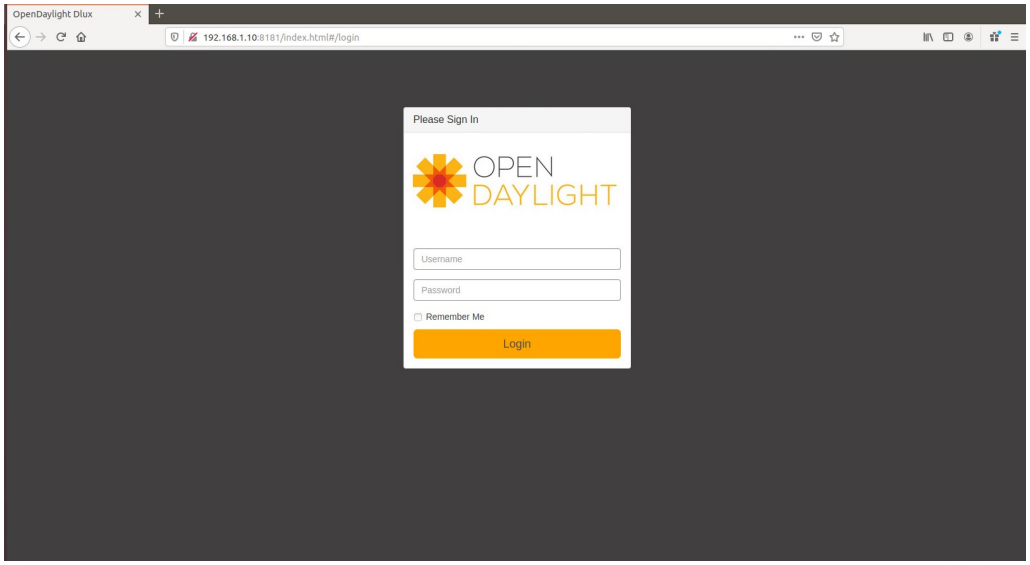
```
odl-mdsal-all
```

2.6.5 OpenDaylight grafické rozhranie

Po inštalácii feature DLUX som rozšíril OpenDaylight o funkciu grafického rozhrania. Grafické rozhranie je dostupné cez webové rozhranie na porte 8181. Pre vstup do grafického rozhrania je potrebné do prehliadača napísať:

```
http://<IP OpenDaylight>:8181/index.html
```

IP adresa OpenDaylight je nastavená na 192.168.1.10. Ako prvá sa zobrazí prihlasovacia obrazovka, ktorá je zobrazená na obr. 2.3. Prihlasovacie meno je admin a heslo admin. DLUX obsahuje viacero položiek ale pre túto prácu bude najpodstatnejšia položka *Topology* v ktorej sa nachádza aktuálny pohľad na topológiu.



Obr. 2.3: Prihlásenie do grafického rozhrania.

2.6.6 Prepojenie OpenDaylight a OVS

Pred tým ako som prepojil OpenDaylight a OVS, musel som im nastaviť IP adresu z rovnakého rozsahu a skontrolovať ich prepojenie. IP adresa pre OpenDaylight je 192.168.1.10/24 a pre OVS 192.168.1.20/24.

Po nastavení IP a otestovaní prepojenia pomocou príkazu ping bolo na rade nastaviť OVS. Po otestovaní funkčnosti prepojenie som vytvoril v OVS bridge na ktorý budú pripojené VM. V OVS som vytvorili bridge br0 pomocou príkazu:

```
$ sudo ovs-vsctl add-br br0
```

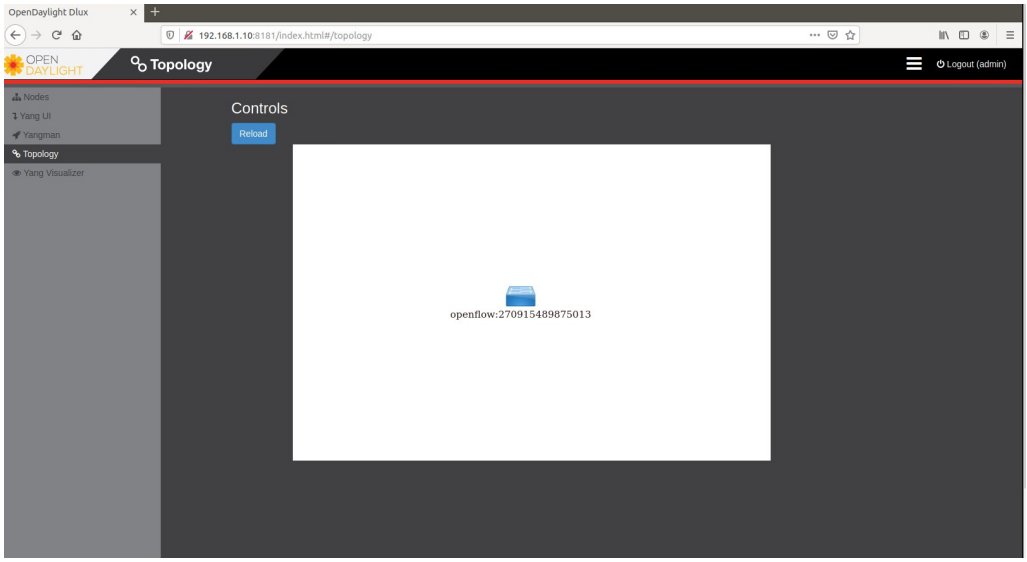
Vytvorený bridge som musel aktivovať pomocou príkazu:

```
$ sudo ip link set br0 up
```

Ako posledný krok som nastavil aby OVS počúval na porte 6653 z adresy 192.168.1.10. Port 6653 slúži na komunikáciu protokolu OpenFlow ako aj port 6633. Toto som nastavil pomocou príkazu:

```
$ sudo ovs-vsctl set-controller br0 tcp:192.168.1.10:6653
```

Samotné nastavenie som skontroloval pomocou grafického rozhrania (obr. 2.4) a pomocou výpisu z OVS (obr. 2.5).



Obr. 2.4: Kontrola prepojenia OpenDaylight a OVS v grafickom rozhraní DLUX.

```
student@student_web:~$ sudo ovs-vsctl show
b1515b36-f950-47f2-9ace-628f768472bc
    Bridge "br0"
        Controller "tcp:192.168.1.10:6653"
            is_connected: true
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.9.5"
student@student_web:~$ _
```

Obr. 2.5: Kontrola prepojenia OpenDaylight a OVS v príkazovom riadku OVS.

2.7 Prvotné riešenie topológie

Prvotné riešenie laboratórnej úlohy bolo vytvoriť na ESXi servery VM, ktorý by bola použitá ako server na ktorom bežia ďalšie VM a tie sú potom prepojené cez OVS. Pre toto riešenie bolo potrebné nainštalovať program ktorý dokázal vytvoriť VM vo VM. Prvotný nápad bolo použitie programu VirtualBox, ale ten nefungoval správne kvôli nedostatočne výkonnému hardvéru. Ako ďalší použitý program na virtuálizáciu bol použitý KVM (Kernel-based Virtual Machine).

2.7.1 Inštalácia KVM

KVM je použité ako hypervizor v mojej topológii. Ako prvý bod inštalácie som zistil či použitý procesor podporuje virtuálizáciu. Toto som zistil pomocou príkazu:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Výstupom tohto príkazu je číselná hodnota. Pokiaľ je číslo väčšie ako 0, tak je virtuálizácia možná.

Ďalší krok je potrebné zistiť ci hardware podporuje KVM virtuálizáciu. Na tento krok je potrebné najprv stiahnuť cpu-checker.

```
$ sudo apt install cpu-checker
```

```
student@student-VirtualBox:~$ sudo kvm-ok
INFO: Your CPU does not support KVM extensions
KVM acceleration can NOT be used
student@student-VirtualBox:~$
```

Obr. 2.6: Kontrola podpory KVM.

Ako je vidieť na obr. 2.6, tak podla výpisu je zrejmé že KVM vizualizácia nieje možná. To nieje pravda. Virtuálizácia je možná len nebude taká rýchla ako v prípade KVM podpory.

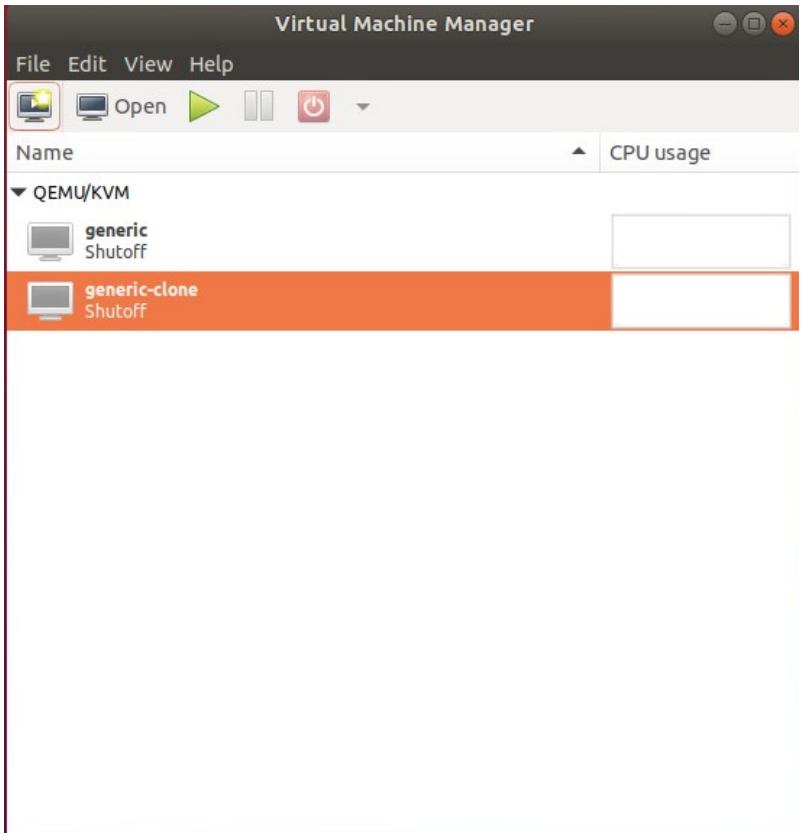
Ako posledný krok som stiahol všetky potrebné balíčky.

```
$ sudo apt install qemu qemu-kvm libvirt-bin  bridge-utils
virt-manager
```

Jeden z balíčkov, ktorý sa stiahol je aj grafické rozhranie *virt-manager*. Inštaláciu VM som realizoval cez grafické rozhranie, to som spustil pomocou príkazu:

```
$ sudo virt-manager
```

Grafické rozhranie je zobrazené na obr. 2.7



Obr. 2.7: KVM grafické rozhranie.

2.7.2 Použitie VM v KVM

Vytvorenie VM v KVM bolo jednoduché a dalo sa realizovať pomocou grafického rozhrania. Operačný systém pre VM bol zvolený Debian pre jeho nenáročnosť na hardware. Systém bol funkčný len mierne spomalený.

2.7.3 Prepojenie KVM VM a OVS

Na prepojenie KVM VM a OVS som vytvoril virtuálne rozhranie tap. Nevýhodou tohto rozhrania bola jeho dočasnosť. Vždy po reštarte bolo potrebné tento port znova vytvoriť.

Tap rozhranie vport1 som vytvoril pomocou príkazu:

```
$ sudo ip tuntap add mode tap vport1
```

Po vytvorení rozhrania som vport1 pridal vo OVS systému pomocou príkazu:

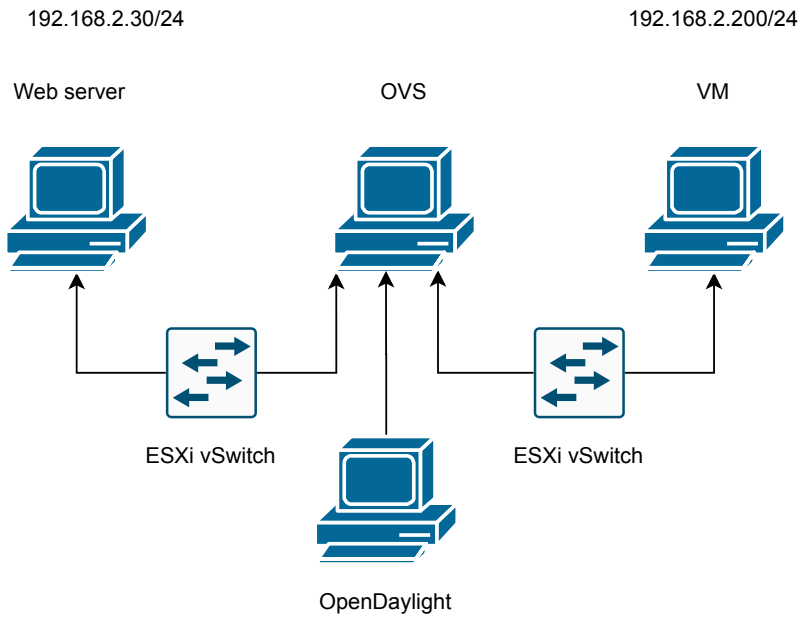
```
$ sudo ovs-vsctl add-port br0 vport1
```

Keď som pridal vport1 do OVS systému bolo pohrebné pripojiť KVM VM do OVS. A v tomto bode nastal problém. Pre nedostatočnú znalosť operačného systému

Ubuntu a programu KVM som sa na tomto mieste zasekol. Po konzultácií s vedúcim mojej práce sme prišli na lepšie riešenie topológie.

2.8 Realizácia laboratórnej úlohy

Ako som už spomenul, tak po konzultácií s vedúcim mojej práce sme prišli na lepšie riešenie topológie. Toto riešenie spočíva s využitím sieťových virtuálnych prepínačov ktoré poskytuje ESXi server. Finálna topológia je zobrazená na obr. 2.8.



Obr. 2.8: Topológia realizovaná v ESXi servery.

2.8.1 Práca s ESXi vSwitch

Aby som zabezpečil úplné oddelenie všetkých virtuálny počítačov v sieti, tak som vytvoril virtuálne prepínače pre každé prepojenie. Vo výsledku som vytvoril 3 virtuálne prepínače, na ktorých som vytvoril skupinu portov, na ktorú som pripojil jednotlivé virtuálne počítače.

Finálne zapojenie vyzeralo tak, že na virtuálnych prepínačoch som vytvoril skupiny portov. Na skupinu portov Web VM som pripojil Web server a OVS, na skupinu portov Host VM som pripojil VM a OVS a v poslednom rade do skupiny portov s názvom Kontrolér som pridal OpenDaylight a OVS. Týmto zapojením som docieli prepojenie Web servera a VM cez OVS virtuálny počítač, ktorý sa v tomto prípade chová ako OpenFlow prepínač.

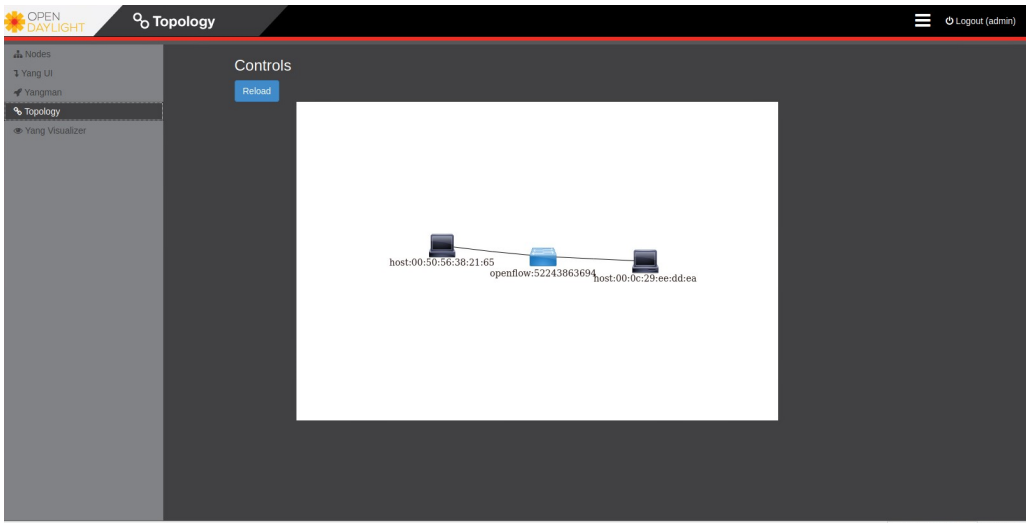
2.8.2 Prepojenie OVS s VM

Ako prvé v tejto topológii som prepojili OVS s VM, ktoré bežia na ESXi servery s využitím vSwitchov, ktoré server ponúka. Aby boli VM oddelené, tak som vytvoril dva vSwitche, ktoré neboli prepojené. Na tieto oddelené vSwitche som pripojil VM, Web server a OVS ako je zobrazené na obr. 2.8.

Aby bolo možne prepojiť VM a Web server musel som pridať do OVS systému na bridge br0 porty, ktoré sú pripojené na vSwitchoch k VM a Web serveru. V mojom prípade to boli porty ens34 a ens38. Pridanie do OVS systému som realizoval pomocou príkazu:

```
$ sudo ovs-vsctl add-port br0 ens34
$ sudo ovs-vsctl add-port br0 ens38
```

V grafickom rozhraní DLUX bude vidieť pripojené VM až po úspešnom realizovaní pingu. Preto som zadal ping na všetkých pripojených zariadeniach a skúsiť ich prepojenie. Pokiaľ je ping úspešný, tak je vidieť kompletná topológia v grafickom rozhraní DLUX (obr. 2.9).



Obr. 2.9: Realizovaná topológia zobrazená v grafickom rozhraní OpenDaylight.

2.9 OpenDaylight OpenFlow Manager (OFM)

Ako už bolo spomenuté, tak rozširujúca aplikácia OFM je použitá na úpravu tokov na OVS. OFM je REST API od firmy Cisco. Features na prepojenie OFM a OpenDaylight sú:

- restconf
- l2switch

- openflowplugin
- yangtools
- mdsal

Inštalácia OFM

Ako prvé som nainštaloval npm a nodejs pomocou príkazu:

```
$ sudo apt-get install -y npm
$ sudo apt-get install -y nodejs
```

Po inštalácií týchto dvoch rozšírení som stiahol samotnú aplikáciu OFM. Aplikácia OFM je dostupná na githube. OFM som stihol pomocou príkazu:

```
$ sudo git clone https://github.com/CiscoDevNet/
OpenDaylight-Openflow-App
```

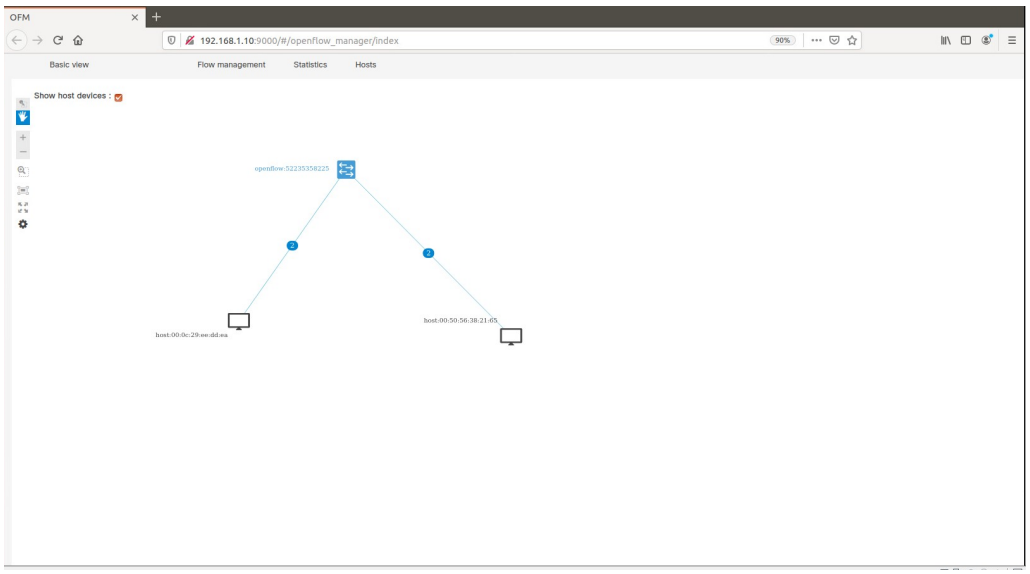
Po stiahnutí som vstúpil do OpenDayLight-OpenFlow-App. V tejto zložke som nainštaloval grunt-cli pomocou príkazu:

```
$ sudo npm install -g grunt-cli
```

Ako posledné som spustil localhost server na porte 9000 pomocou príkazu:

```
$ sudo grunt
```

Správne nainštalovanie OFM som skontroloval v prehliadači na adrese 192.168.1.10:9000 (obr. 2.10).



Obr. 2.10: Grafické rozhranie OFM.

2.9.1 Úprava tokov cez OFM

Na úpravu tokov slúži možnosť *Flow management*, ktorá sa nachádza v hornej lište. Toky, ktoré sa tam nachádzajú sú vygenerované OpenDaylight feature l2switch, hneď po pripojení OVS a ODL. Tie vygenerované toky sa nedajú upravovať, preto som v mojej práci tieto toky zmazal a vytvoril som si vlastné. Ako príklad spomeniem len pár tokov. Zbytok tokov bude v prílohe *Laboratórny návod* ako ukážka pre študenta alebo samostatná úloha pre študenta.

Tok na povolenie ARP a zahodenie komunikácie

Ako som už spomenul, tak laboratórna úloha bude postavená na tom, že študent zakáže celkovú komunikáciu a pomocou tokov bude povoľovať jednotlivé typy komunikácie. Na to aby fungovala celková komunikácia, tak je potrebné nastaviť ARP protokol. Ja som ARP protokol nastavil pomocou toku, v ktorom som zvolil pole zhody Ethernet type a v ňom som zvolil hodnotu 0x0806. Ako akciu som zvolil Flood, čo zabezpečuje odoslanie ARP na pripojené zariadenia. Pre zahadzovanie komunikácie som nevybral žiadne pole zhody a len som zvolil akciu Drop. Finálne toky vyzerali takto:

				Match	Action
Polia	Table	ID	Priority	Ethernet type	
Hodnoty Toku	0	ARP	1	0x0806	Flood
Hodnoty Toku	0	Drop	0		Drop

Tab. 2.1: Nastavenie pre toky Drop a ARP.

Toky na povolenie ICMP

Aby bolo možne realizovať napríklad ping medzi počítačmi, tak je potrebné aby boli toky nastavené z každej strany komunikácie. Pre povolenie ICMP som zvolil tieto polia zhody:

- In port
- Ethernet type
- IP protokol

Polia pre Ethernet type obsahovalo hodnotu 0x0800 a pole pre IP protokol hodnotu 1. Pole pre In port obsahovalo vstupný port. Keďže sú v mojej topológii dva počítače, tak In port obsahovalo vždy inú hodnotu portu.

Ako akcia na vykonanie pri zhode som nastavil Output port s veľkosťou 9999. Finálne toky vyzerali takto:

				Match			Action	
Polia	Table	ID	Priority	In port	Ethernet type	IP protocol	Output port	Maximum length
Hodnoty Toku	0	Ping1	2	4	0x0800	1	5	9999
Hodnoty Toku	0	Ping2	2	5	0x0800	1	4	9999

Tab. 2.2: Nastavenie pre ICMP toky.

Pole Table značí tabuľku, do ktorej sa uloží tok. Pri vytváraní tokov som vždy použil tabuľku 0. Pole ID značí pomenovanie toku, podľa ktorého sa dá tok hľadať v OFM. Hodnota pola Priority udáva prioritu toku. Čim väčšie číslo, tým väčšia priorita. IN port udáva, na ktorom porte sa budú hľadať polia zhody. Polia zhody Ethernet type a IP protokol už potom podrobne určujú, o aký typ komunikácie ide. Akcia Output port určuje, na ktorý post sa pošle komunikácia ktorá sa zhoduje s poliami zhody.

V laboratórnej úlohu bude mať študent za úlohu upraviť tento tok aby ping prechádzal len jedným smerom.

Nastavené toky sa dajú pozrieť aj v OVS (obr. 2.11).

```
student@student_ovs:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows br0
 cookie=0x0, duration=13.812s, table=0, n_packets=0, n_bytes=0, priority=2,icmp,in_port=ens38 action
s=output:ens34
 cookie=0x0, duration=13.808s, table=0, n_packets=0, n_bytes=0, priority=2,icmp,in_port=ens34 action
s=output:ens38
 cookie=0x0, duration=13.812s, table=0, n_packets=0, n_bytes=0, priority=1,arp actions=FLOOD
 cookie=0x0, duration=13.808s, table=0, n_packets=0, n_bytes=0, priority=1 actions=drop
student@student_ovs:~$ _
```

Obr. 2.11: Výpis tokov v OVS.

Toky v laboratórnej úlohe

V laboratórnej úlohe bude mať študent k dispozícií základne toky na povolenie ARP, zahadzovanie komunikácie a na povolenie ICMP správ. Úloha pre študenta bude vytvoriť toky na povolenie SSH, HTTP a pingu len z VM na Web server. Po nastavení týchto tokov bude mať za úlohu povoliť ping len na 10 sekúnd. Po tejto dobe sa tok na povolenie pingu sám zmaže. Ako posledná úloha bude nastavenie toku na zahadzovanie komunikácie s prioritou väčšou ako všetky ostatné toky. Pri prezentovaní výsledkov vyučujúcemu, bude musieť študent odpovedať na otázky, ktoré sa nachádzajú na konci laboratórnej úlohy.

Záver

Cieľom tejto práce, ktorá bola rozdelená na semestrálnu prácu a bakalársku prácu bolo analyzovať súčasné možnosti SDN pre cloud computing, vytypovať vhodný SDN kontrolér a realizovať laboratórnu úlohu, na ktorej sa budú študenti oboznamovať s touto technológiu.

V teoretickej časti je popísaný princíp fungovania SDN, popis funkčnosti protokolu OpenFlow vo verzií 1.0, z ktorej vychádzajú ostatné verzie tohto protokolu, stručný popis niektorých SDN kontorlérov, výhody SDN v prípade QoS, závislosť SDN na protokole Netconf, modelovací jazyk Yang, vzťah medzi SDN a NFV a virtuálizácia v oblasti dátových centier.

Praktická časť obsahuje podrobnejší popis zvoleného kontroléra. Zvolený bol kontrolér OpenDaylight pre jeho dobré vlastnosti v oblasti cloud computingu a dátových centier. Ďalej sa tu nachádza popis použitého virtuálneho prepínača OVS. V časti Topológia siete je vysvetlený spôsob realizácie zapojenia a popis toho k čomu je topológia realizovaná. Zbytok práce je zameraný na inštaláciu softvérových komponentov na ESXi servery, na prácu s manažérom tokov OFM a podrobný popis funkčnosti základných tokov, ktoré sú použité v laboratórnej úlohe.

Jedna kapitola v praktickej časti je zameraná na opísanie prvotného riešenia topológie. Toto riešenie bolo postavené na vytvorení VM v inej VM, ktorá fungovala na ESXi servery. Pri realizácií tohto riešenia nastalo pár problémov ako rýchlosť a stabilita VM, práca so sieťovou stránkou v KVM a prepojenie KVM VM s OVS. Po konzultácií s vedúcim práce sa topológia siete prepracovala na jej finálnu podobu, v ktorej sa používajú virtuálne prepínače poskytované ESXi serverom.

Vecným výstupom tejto práce je laboratórna úloha a laboratórny návod. Laboratórna úloha je zameraná na oboznámenie študenta s vytváraním a modifikovaním tokov v SDN sieti. Zo začiatku bude študent oboznámený so základným tokmi na zahadzovanie komunikácie Drop, na povolenie protokolu ARP a na povolenie ICMP správ. V ďalšej časti bude mať študent za úlohu vytvoriť toky na povolenie SSH, HTTP a povolenie ping správy z jedného zariadenia na druhé. Na realizovanie týchto úkonov bude musieť využiť svoje znalosti z oblasti komunikačných portov, Ethernet rámca, IP packetu a ICMP správ ping. V poslednom rade sa študent oboznámi so základnými príkazmi v operačnom systéme Linux.

Literatúra

[1] *Performance evaluation of live virtual machine migration in SDN-enabled cloud data centers.* ScienceDirect [online]. 2019 [cit. 2019-12-21]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S074373151830474X#!>.

[2] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 1 ISBN 978-0-12-804555-8.

[3] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 9 ISBN 978-0-12-804555-8.

[4] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 100-103. ISBN 978-0-12-804555-8.

[5] HowtoForge [online]. [cit. 29. 11. 2019]. Dostupné z: <https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/>.

[6] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 101. ISBN 978-0-12-804555-8.

[7] *A combined Network Access Control and QoS scheme for Software Defined Networks.* IEEE [online]. Italy: IEEE, 2019 [cit. 2019-12-10]. Dostupné z: <https://.ieee.org/document/8725732>.

[8] *Quality of Service (QoS) in Software Defined Networking (SDN): A survey.* ScienceDirect [online]. USA: University Indianapolis, 2016 [cit. 2019-12-10]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S1084804516303186>.

[9] *Optimal placement of virtual network functions in software defined networks: A survey.* ScienceDirect [online]. Turecko: Gazi University, 2019 [cit. 2019-12-15]. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S1084804519302760#fig2>.

[10] NADEAU, Thomas D. *Network function virtualization. Cambridge, MA: Elsevier*, 2016. ISBN 9780128021194.

[11] The Control Plane. NADEAU, Thomas D. a Ken GRAY. *SDN: software defined networks.* Beijing: O'Reilly, [2013], s. 11–15. ISBN 978-1-449-34230-2.

[12] Data Plane. NADEAU, Thomas D. a Ken GRAY. *SDN: software defined networks.* Beijing: O'Reilly, [2013], s. 16-18. ISBN 978-1-449-34230-2.

[13] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 90. ISBN 978-0-12-804555-8.

[14] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 92. ISBN 978-0-12-804555-8.

[15] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 93. ISBN 978-0-12-804555-8.

[16] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 94. ISBN 978-0-12-804555-8.

[17] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 95. ISBN 978-0-12-804555-8.

[18] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 95–97. ISBN 978-0-12-804555-8.

[19] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 97–99. ISBN 978-0-12-804555-8.

[20] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 103–.104 ISBN 978-0-12-804555-8.

[21] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 105. ISBN 978-0-12-804555-8.

[22] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 105–107. ISBN 978-0-12-804555-8.

[23] ONF [online]. ONF, 2019 [cit. 2019-12-15]. Dostupné z: <https://www.opennetworking.org/onos/?utm_referrer=https%3A%2F%2Fwww.google.com%2F>.

[24] Ryu [online]. Nippon Telegraph, 2014 [cit. 2019-12-17]. Dostupné z: <https://ryu.readthedocs.io/en/latest/getting_started.html#what-s-ryu>.

[25] OpenDaylight Documentation [online]. OpenDaylight Project, 2018 [cit. 2019-12-18]. Dostupné z: <https://docs.opendaylight.org/en/stable-sodium/getting-started-guide/introduction.html>.

[26] Maestro [online]. [cit. 2019-12-18]. Dostupné z: <https://code.google.com/archive/p/maestro-platform/>.

[27] Project Floodlight [online]. Project Floodlight, 2019 [cit. 2019-12-18]. Dostupné z: <http://www.projectfloodlight.org/floodlight/>.

[28] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 151. ISBN 978-0-12-804555-8.

[29] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 172–173. ISBN 978-0-12-804555-8.

[30] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 173. ISBN 978-0-12-804555-8.

[31] GORANSSON, Paul, Chuck BLACK a Timothy CULVER. *Software defined networks: a comprehensive approach. Second edition.* Singapore: Morgan Kaufmann, [2017], s. 32. ISBN 978-0-12-804555-8.

[32] *Open vSwitch: Why Open vSwitch [online]*, 2020. Dostupné z: <https://github.com/openvswitch/ovs/blob/master/Documentation/intro/why-ovs.rst>

[33] *VMware: VMware vSphere Documentation [online]*, 2020. Dostupné z: <https://docs.vmware.com/en/VMware-vSphere/index.html>

[34] *OpenDaylight Documentation: DLUX [online]*, 2018. Dostupné z: <https://docs.opendaylight.org/en/stable-oxygen/developer-guide/dlux.html?highlight=DLUX>

[35] *OpenDaylight Documentation: DLUXOpenFlowPlugin Project [online]*, 2018. Dostupné z: <https://docs.opendaylight.org/en/stable-oxygen/release-notes/projects/openflowplugin.html?highlight=Openflowplugin>

[36] *OpenDaylight Documentation: L2Switch [online]*, 2018. Dostupné z: <https://docs.opendaylight.org/en/stable-oxygen/release-notes/projects/l2switch.html?highlight=l2switch>

[37] *OpenDaylight Documentation: YANG Tools [online]*, 2018. Dostupné z: <https://docs.opendaylight.org/en/stable-nitrogen/release-notes/projects/yangtools.html>

[38] *OpenDaylight Documentation: MD-SAL Overview [online]*, 2018. Dostupné z: <https://docs.opendaylight.org/en/stable-oxygen/developer-guide/controller.html#mdsal-dev-guide>

Zoznam symbolov, veličín a skratiek

ACK	Acknowledgement
ACL	Access Control List
API	Application Programming Interface
ARP	Address Resolution Protocol
ASIC	Application Specific Integrated Circuit
CLI	Command Line Interface
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
DLUX	OpenDaylight User Experience
DNS	Domain Name System
EVB	Edge Virtual Bridge
EVB	Edge Virtual Bridge
FIB	Forwarding Information Base
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPv4	Internet Protocol version 4
KVM	Kernel-based Virtual Machine
MD-SAL	Model-Driven Service Abstraction Layer
Netconf	Network configuration protocol
NFV	Network Function Virtualization
OFM	OpenDaylight OpenFlow Manager
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OSPF	Open Shortest Path First
OVS	Open vSwitch
OVSDB	Open vSwitch Database Management Protocol
REST	Representational state transfer
RIB	Routing Information Base
RPC	Remote Procedure Calls
SFC	Service Function Chaining
SLA	Service Level Agreement
SDN	Software Defined Network
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SYN	Synchronization

TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOS	Type of Service
TTL	Time to Live
UDP	User Datagram Protocol
VEPA	Virtual Edge Port
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
XML	Extensible Markup Language
YANG	Yet Another Next Generation

Zoznam príloh

A Príklad tokov v JSON formáte **52**
A.1 Príklad toku Drop 52
A.2 Príklad toku ARP 52
A.3 Príklad toku pre povolenie ICMP 53

B Laboratórny návod **56**
B.1 Ciel úlohy 56
B.2 Pracovisko 56
B.3 Úlohy 56
B.4 Teoretický úvod 56
B.4.1 Dôvod vzniku Softvérovo-definovaných sietí 56
B.4.2 Princíp fungovania Softvérovo-definovaných sietí 57
B.4.3 Architektúra SDN 58
B.4.4 Open vSwitch 59
B.4.5 OpenDaylight 60
B.5 Postup riešenia 60
B.5.1 Topológia 60
B.5.2 Spustenie potrebných aplikácií 61
B.5.3 Úprava tokov 62
B.5.4 Samostatná práca 64
B.6 Otázky 65
B.7 Upratanie pracoviska 65

A Príklad tokov v JSON formáte

A.1 Príklad toku Drop

```
{
"flow": [
    {
        "table_id": 0,
        "id": "Drop",
        "priority": 0,
        "hard-timeout": 0,
        "idle-timeout": 0,
        "instructions": {
            "instruction": [
                {
                    "order": 0,
                    "apply-actions": {
                        "action": [
                             {
                            "order": 0,
                            "drop-action": {}
                                }
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

A.2 Príklad toku ARP

```
{
"flow": [
    {
        "table_id": 0,
        "id": "ARP",
```

```
        "priority": 1,
        "hard-timeout": 0,
        "idle-timeout": 0,
        "match": {
            "ethernet-match": {
                "ethernet-type": {
                    "type": 2054
                    }
                }
            },
            "instructions": {
            "instruction": [
                {
                "order": 0,
                "apply-actions": {
                    "action": [
                        {
                        "order": 0,
                        "output-action": {
                        "output-node-connector": "FLOOD"
                                    }
                                }
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

A.3 Príklad toku pre povolenie ICMP

```
{
"flow": [
    {
        "table_id": 0,
        "id": "ICMP1",
```

```
            "priority": 2,
            "hard-timeout": 0,
            "idle-timeout": 0,
            "match": {
                "in-port": "openflow:52235358225:4",
                "ethernet-match": {
                    "ethernet-type": {
                        "type": 2048
                    }
                },
                "ip-match": {
                    "ip-protocol": 1
                    }
                },
            "instructions": {
                "instruction": [
                    {
                        "order": 0,
                        "apply-actions": {
                            "action": [
                                {
                                "order": 0,
                                "output-action": {
                                "output-node-connector": "5",
                                "max-length": 9999
                                        }
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
        ]
}

{
"flow": [
    {
```

```
            "table_id": 0,
            "id": "ICMP2",
            "priority": 2,
            "hard-timeout": 0,
            "idle-timeout": 0,
            "match": {
                "in-port": "openflow:52235358225:5",
                "ethernet-match": {
                    "ethernet-type": {
                        "type": 2048
                    }
                },
                "ip-match": {
                    "ip-protocol": 1
                    }
                },
                "instructions": {
                "instruction": [
                    {
                        "order": 0,
                        "apply-actions": {
                            "action": [
                                {
                                    "order": 0,
                                    "output-action": {
                                    "output-node-connector": "4",
                                    "max-length": 9999
                                        }
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
```

B Laboratórny návod

B.1 Ciel úlohy

Cieľom tejto laboratórnej úlohy je oboznámenie s technológiou Softvérovo-definovaných sieti (SDN), s protokolom OpenFlow, SDN kontorolérom OpenDaylight a virtuálnym prepínačom Open vSwitch (OVS). V praktickej časti laboratórnej úlohy je za úlohu ovládať dátové toky na OVS pomocou SDN kontroléra OpenDaylight.

B.2 Pracovisko

V tejto laboratórnej úlohe budete pristupovať na ESXi server, na ktorom sú pripravené virtuálne počítače na realizáciu tejto úlohy. Na servery sa nachádzajú:

- OpenDaylight (Virtuálny počítač, na ktorom je nainštalovaný kontrolér OpenDaylight)
- OVS (Virtuálny Ubuntu server, na ktorom je nainštalovaný virtuálny prepínač OVS)
- Web server (Virtuálny Ubuntu server, na ktorom je nainštalovaný web server a SSH server)
- VM (Virtuálny počítač, ktorý je pripravený na pripojenie k Web serveru)

B.3 Úlohy

- Oboznámenie sa s OpenFlow konfiguráciou OVS
- Oboznámenie sa s modifikáciou dátových tokov
- Vytvorenie vlastných tokov

B.4 Teoretický úvod

B.4.1 Dôvod vzniku Softvérovo-definovaných sietí

Postupom času ako sa začal zvyšovať výkon zariadení a začali sa využívať cloudové služby, to viedlo k logickému zvyšovaniu nárokov na počítačové siete. Ako na ich komplexnosť tak aj veľkosť prenášaných dát. Problém nastáva pokiaľ klasickú sieť chceme rozšíriť o nové zariadenie. Tento proces býva časovo dosť náročný z pohľadu konfigurácie zariadenia a troubleshutingu danej konfigurácie. Aby sme tomuto problému predišli, začala sa riadiaca časť siete zjednocovať na jedno zariadenie.

Technológia, ktorá toto zabezpečuje sa nazýva SDN. Príchodom SDN prišli nové inovácie, ktoré uľahčujú spravu a konfiguráciu siete. Jedna z inovácií je rozdelenie riadiacej a dátovej roviny sieťových zariadení. Týmto rozdelením sa dosiahli to, že o spravu a kontrolu siete sa stará centrálna jednotka, ktorá sa nazýva kontrolér. SDN našlo uplatnenie ako v centralizácií a virtuálizácií lokálnych sietí, tak aj v oblasti cloud computing.

B.4.2 Princíp fungovania Softvérovo-definovaných sietí

Jednou z myšlienok využitia SND je v oddelení riadiacich a dátových rovinách sieťových zariadení. Takéto oddelenie dáva prevádzkovateľovi siete určité výhody, pokiaľ ide o centralizované alebo semi-centralizované distribučné možnosti. Tiež má potenciálnu ekonomickú výhodu založenú na schopnosti zjednotiť sa na jednom alebo viacerých miestach, čo umožňuje jednoduchšiu konfiguráciu, čo v konečnom dôsledku vychádza lacnejšie ako klasický spôsob.

Oddelenie kontrolnej a dátovej roviny je jedna zo základných vlastností SDN.

Kontrolná rovina

Kontrolná rovina zaradí lokálne dáta, ktoré použije na vytvorenie vstupov v smerovacej tabuľke, ktoré využije dátová rovina na smerovanie toku dát na vstupné alebo výstupné porty na zariadení. Súbor, do ktorého sa ukladajú informácie o topológii siete sa nazýva „routing information base“ (RIB). Táto databáza sa udržuje konzistentná (bez slučiek) pomocou výmeny informácií medzi jednotlivými kontrolnými rovinami v sieti. Vstupy smerovacej tabuľky sa nazývajú „forwarding information base“ (FIB) a zrkadlia sa medzi kontrolnú a dátovú rovinu zariadenia. FIB sa naprogramuje, keď sa RIB považuje za konzistentný a nemenný. Na vykonanie tejto úlohy musí riadiaca jednotka vytvoriť pohľad na sieťovú topológiu, ktorá spĺňa určité obmedzenia. Tento pohľad na sieť sa dá naprogramovať ručne, naučiť na základe pozorovania siete alebo vytvoriť na základe zozbieraných informácií prostredníctvom komunikácie s inými príkladmi kontrolných rovín, čo môže byť prostredníctvom použitia jedného alebo viacerých smerovacích protokolov, manuálneho programovania alebo kombináciou oboch.

Dátová rovina

Dátová rovina spracováva prichádzajúce datagramy (na drôte, optickom vlákne alebo bezdrôtovo) prostredníctvom série operácií na úrovni pripojení, ktoré tieto datagramy zachytávajú a vykonávajú základnú kontrolu. Správny datagram sa spracováva v dátovej rovine prostredníctvom vyhľadávaní v tabuľke FIB, ktorá je na-

programovaná skôr kontrolnou rovinou. Toto sa niekedy označuje, ako rýchla cesta spracovávania paketov, pretože nepotrebujú už žiadne ďalšie dodatočné informácie, okrem tých, ktoré sú obsiahnuté vo FIB tabuľke. Výnimka nastáva, keď paket nespĺňuje tieto pravidlá, ako napríklad, keď sa zistí neznámy cieľ, tak sa tieto dáta odošlú do procesora trasy, kde ich kontrolná rovina spracuje s použitím RIB tabuľky

B.4.3 Architektúra SDN

SDN sa vo všeobecnosti skladá z troch vrstiev:

- Aplikačná vrstva
- Riadiaca vrstva
- Vrstva infraštruktúry

Aplikačná vrstva

Otvorená oblasť, ktorá umožňuje vývoj inovatívnych aplikácií s využívaním všetkých informácií o sieťovej topológii. Je niekoľko druhov aplikácií, ktoré je možno vyvíjať zo zameraním na riešenie problémov so sieťou, sieťová politika, bezpečnosť, aplikácie súvisiace s automatizáciou siete, sieťová konfigurácia a správa a monitorovanie siete. Tieto aplikácie môžu ponúknuť rôzne komplexné riešenia pre podnikové siete a siete dátových centier v reálnom čase. Sieťový sprostredkovatelia ponúkajú vlastné riešenia svojich SDN aplikácií

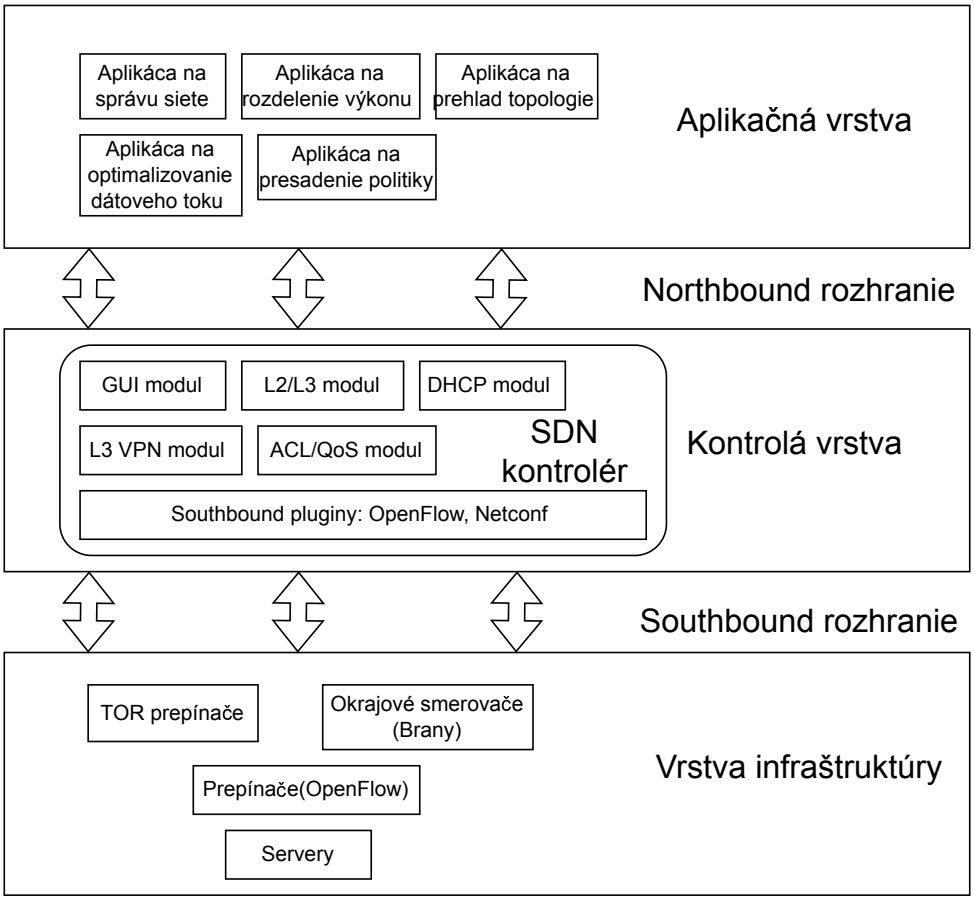
Riadiaca vrstva

Je miesto, kde sa uplatňuje kontrolná rovina a kontrolu nad sieťou zabezpečuje SDN kontrolér. Toto je oblasť, kde každý sieťový sprostredkovateľ pracuje na tom, aby prišiel s vlastnými produktami pre SDN kontrolér alebo framework. V tejto vrstve je v kontroléri napísaných veľa administratívnych logík, aby bolo možné udržiavať rôzne typy informácií o sieti, podrobnosti o stave, podrobnosti o topológii, štatistické údaje a ďalšie. Kontrolná rovina oddeľuje dva typy rozhraní:

- **Southbound rozhranie** : je určený na komunikáciu s nižšou vrstvou infraštruktúry sieťových prvkov a vo všeobecnosti je realizovaný pomocou southbound protokolov - OpenFlow, Netconf, OVSDB...
- **Northbound rozhranie** : je určený na komunikáciu s hornou aplikačnou vrstvou a vo všeobecnosti je realizovaný prostredníctvom REST API SDN kontroléra.

Vrstva infraštruktúry

Skladá sa z rôznych sieťových zariadení, ktoré tvoria základnú sieť na smerovanie. Môže to byt sada prepínačov a smerovačov v dátových centrách. Táto vrstva je fyzická, na ktorú sa prostredníctvom riadiacej vrstvy stanovila virtuálizácia siete (kde SDN kontrolér riadi základnú fyzickú sieť).



Obr. B.1: SDN architektúra

B.4.4 Open vSwitch

V prípade hypervizorov založených na Linuxe sa na premostenie komunikácie medzi VM a okolitým svetom môže použiť rýchli a spoľahlivý L2 prepínač (Linux bridge). Toto riešenie nieje zlé ale v prípade použitia v multi-serverovej virtuálizácií nieje vhodný. Tieto prostredia sa často vyznačujú vysoko dynamickými koncovými bodmi, udržiavaním logických abstrakcií a v niektorých prípadoch integráciu s prepínacím hardware. V tomto prípade je vhodnejšie použiť OVS.

B.4.5 OpenDaylight

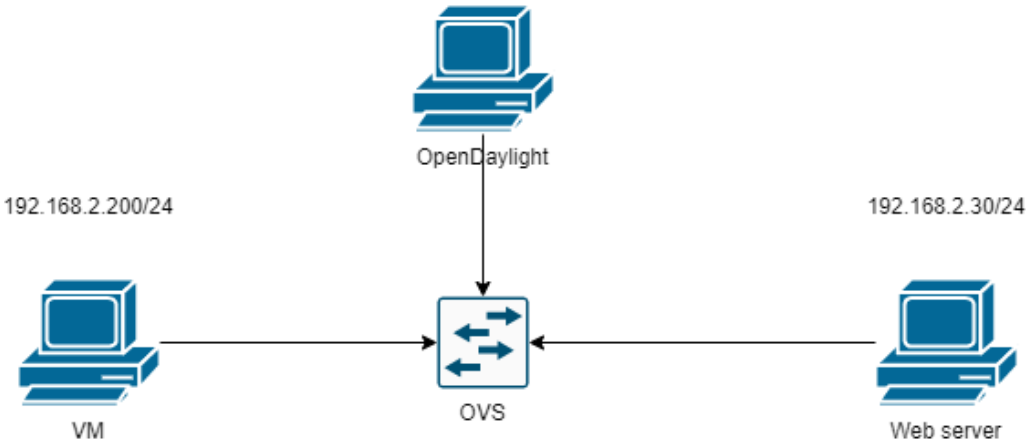
Hlavné rozdiely v OpenDaylight SDN v porovnaní s inými možnosťami SDN sú:

- Architektúra mikroprocesorov, v ktorej „mikroservis“ je konkrétny protokol alebo služba, ktorú chce používateľ povoliť v rámci svojej inštalácie OpenDaylight.
  - Doplnok, ktorý poskytuje pripojenie k zariadeniam prostredníctvom OpenFlow protokolov.
  - Platformová služba, ako napríklad overenie, autorizácia a účtovníctva.
  - Sieťová služba poskytujúca pripojenie VM pre OpenStack.
- Podpora širokého a rastúceho množstva sieťových protokolov: OpenFlow, BGP, Netconf, SNMP...
- MD-SAL (Model Driven Service Abstraction Layer). Yang modely zohrávajú v OpenDaylight kľúčovú úlohu a používajú sa pre:
  - Vytváranie datastore schémat.
  - Vytváranie aplikácií.
  - Automatické generovanie kódu.

OpenDaylight je napísaný v programovacom jazyku JAVA a je vhodný na cloudové riešenia

B.5 Postup riešenia

B.5.1 Topológia



Obr. B.2: Topológia laboratórnej úlohy.

B.5.2 Spustenie potrebných aplikácií

Vyhladajte aplikáciu VMware (bude cez vyhľadávanie alebo ju nájdite na pracovnaj ploche). Pripojte sa na ESXi server (na ľavej strane v lište IP adresa 10.100.108.253), na ktorom sú už pripravené VM a všetky spustite. Prihlasovacie mená na VM sú **student** a heslo **student**. Na VM OVS je potrebné skontrolovať, či sú všetky potrebné rozhrania aktívne. To sa dá skontrolovať pomocou príkazu:

```
$ ip a
```

Po zadaní príkazu sa zobrazí tabuľka rozhraní v systéme Ubuntu. Je potrebné aby boli všetky rozhrania aktívne. Rozhranie sa dá aktivovať pomocou príkazu:

```
$ sudo ip link set "rozhranie" up
```

Skontrolujte či ste aktivovali rozhrania. Po nastavení rozhraní skontrolujte nastavenie OVS pomocou príkazu:

```
$ sudo ovs-vsctl show
```

Konfigurácia by mala vyzerať ako na obr. B.3.

```
student@student_ovs:~$ sudo ovs-vsctl show
b1515b36-f950-47f2-9ace-628f768472bc
    Bridge "br0"
        Controller "tcp:192.168.1.10:6653"
        Port "ens38"
            Interface "ens38"
        Port "ens34"
            Interface "ens34"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.9.5"
student@student_ovs:~$ _
```

Obr. B.3: Nastavenie OVS

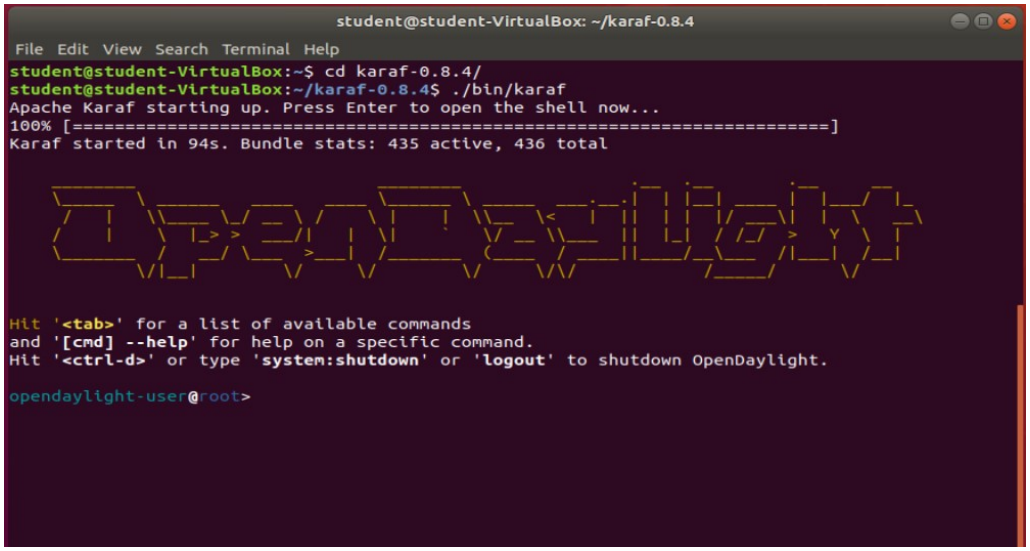
Pokiaľ je OVS nastavený ako podla obr. B.3 môžete tento krok preskočiť. Pokiaľ nieje nastavíte ho podla príkazov:

```
$ sudo ovs-vsctl add-port br0 ens34
$ sudo ovs-vsctl add-port br0 ens38
$ sudo ovs-vsctl add-br br0
$ sudo ovs-vsctl set-controller br0 tcp:192.168.1.10:6653
```

Prepnite sa na VM OpenDaylight. To sa v ľavej lište nahádza terminál. Pre pustenie kontroléru je potrebné do terminálu zadať tieto príkazy:

```
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
$ cd karaf-0.8.4/
$ ./bin/karaf
```

Po zadaní týchto príkazov by sa vám mal začať púšťať OpenDaylight (obr. B.3). Tento terminál nechajte otvorený (pokiaľ by ste ho zatvorili, tak sa vypne Open-



Obr. B.4: Spustenie OpenDaylight

Daylight a bude ho musieť pustiť znova) a otvorte si ďalší. V tomto okne napíšte príkazy:

```
$ cd OpenDaylight-Openflow-App/
$ sudo grunt
```

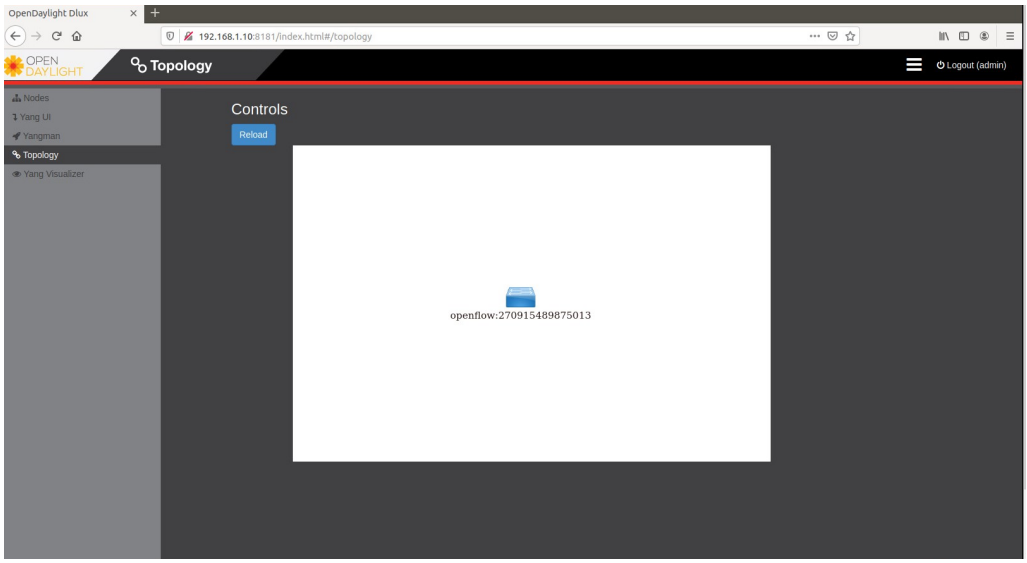
Pomocou týchto príkazov vstúpite do adresára OFM kde spustíte server na porte 9000. V tento moment v tento moment by mal byť prepojený OpenDaylight a OVS. Prepojenie môžete skontrolovať napríklad v grafickom rozhraní DLUX dostupného v prehliadači. Do prehliadača zadajte:

```
http://192.168.1.10:8181/index.html
```

Zobrazí sa vám grafické rozhranie OpenDaylight. Prihlasovacie meno je **admin** a heslo **admin**. Na ľavej strane v položke Topology sa vám zobrazí pripojený OVS (obr. B.5). V tento moment ste schopný realizovať ping medzi VM a Web serverom. Otestujte prepojenie. Po úspešnom pingu uvidíte pripojené počítače aj v grafickom rozhraní DLUX.

B.5.3 Úprava tokov

V prehliadači otvorte nové okno a doňho zadajte:



Obr. B.5: Kontrola prepojenia OpenDaylight a OVS v grafickom rozhraní DLUX.

```
http://192.168.1.10:9000
```

Zobrazí sa vám grafické rozhranie OFM. Ako prvé môžete vidieť jeden prepínač bez pripojených VM. To zmeníte zaškrtnutím **Show host devices** v ľavom hornom rohu. Polial ste postupovali podla návodu a ping medzi VM úspešný, tak uvidíte v záložke hosts tabuľku pripojených hostov. Prepnite sa do záložky **Flow management**. Táto záložka bude slúžiť na úpravu tokov na pripojenom OVS. V spodnej časti sa nachádzajú už vygenerované toky pomocou l2switch feature, ktorá je nainštalovaná v OpenDaylight. Všetky toky označte a zmažte (v ľavom rohu tabuľky sa nachádza pero na vytvorenie tokov, krížik na zmazanie tokov a šípka na znovu načítanie tokov).

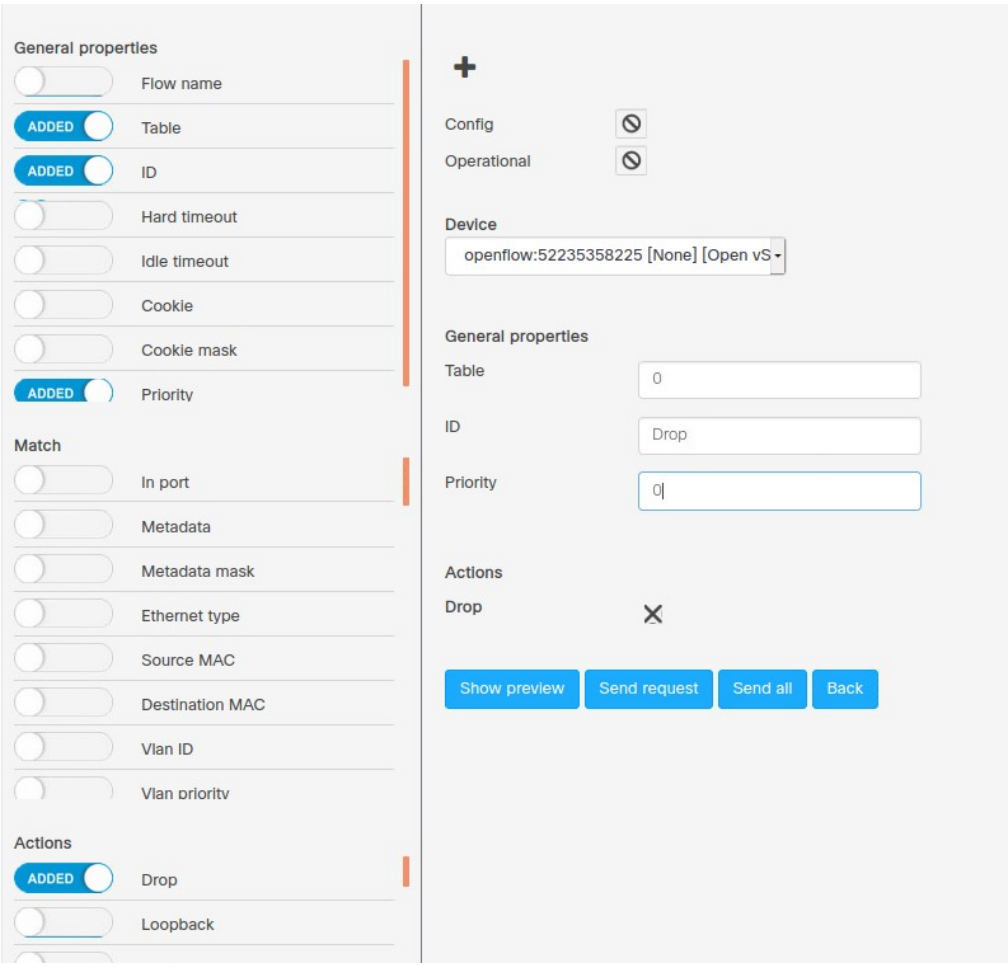
Prvé toky na vytvorenie budú Drop na zahodenie komunikácie a ARP. Vo Flow management klikneme na ikonku pera a otvori sa nám okno na vytvorenie nového toku. Tok nastavte podla obr. B.6 Tok odošlete do zariadenia kliknutím na možnosť **Send request**.

V prípade toku ARP nastavte tabulku 0 a prioritu na 1. V možnostiach **Match** vyberte možnosť **Ethernet type**. Do políčka Ethernet type napíšte hodnotu 0x0806 a za **Actions** zvoľte **Flood**. Tok odošlite (v prípade keď sa vám tok neodošle, tak ho skúste poslať znova, ak nepomôže ani to, tak skontrolujte správnosť nastavenia toku).

Odoslané toky skontrolujte v OVS pomocou príkazu:

```
$ sudo ovs-ofctl -O OpenFlow13 dump-flows br0
```

Ďalšie toky budú na povolenie ICMP správ medzi zariadeniami. v tomto prípade budú potrebné dva toky. Jeden tok od VM ku Web serveru a druhý od Web serveru



Obr. B.6: Nastavenie toku pre zahadzovanie komunikácie.

ku VM. Toky nastavte podla tab. B.1.

				Match			Action	
Polia	Table	ID	Priority	In port	Ethernet type	IP protocol	Output port	Maximum length
Hodnoty Toku	0	Ping1	2	4	0x0800	1	5	9999
Hodnoty Toku	0	Ping2	2	5	0x0800	1	4	9999

Tab. B.1: Nastavenie ICMP tokov.

Pokiaľ ste všetko nastavili správne, tak by mal byť ping medzi VM a Web serverom úspešný.

B.5.4 Samostatná práca

V tejto časti budete mať za úlohu vytvoriť toky samostatne. Toky, ktoré budete musieť vytvoriť sú:

- Toky na povolenie SSH medzi VM a Web serverom.
- Toky na povolenie HTTP medzi VM a Web serverom.

- Povoľte ping len na 10 sekúnd (využite možnosť Hard timeout).
- Toky na povolenie pingu z VM na Web server a zakázanie pingu z Web serveru na VM.
- Tok na zahadzovanie prevádzky s prioritou väčšou ako ostaté toky.

Na realizovanie týchto tokov budete potrebovať údaje z tab. B.2, tab. B.3, tab. B.4. Na spustenie SSH prepojenia medzi VM a Web serverom zadajte tento príkaz:

```
$ sudo ssh student@192.168.2.30
```

Ethernet type	Protokol
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

Tab. B.2: Tabuľka Ethernet type.

Číslo protokolu	Protokol
1	ICMP
6	TCP
17	UDP

Tab. B.3: Tabuľka IP protokolov.

Typ	Význam
0	Echo reply (ping)
8	Echo request (ping)

Tab. B.4: Tabuľka ICMP typov.

B.6 Otázky

- Čo znamená Hard timeout?
- Čo znamená priority?
- Ktorá feature OpenDaylight generuje základné toky?

B.7 Upratanie pracoviska

Ako posledný bod zmažte všetky vaše vytvorené toky a vypnite virtuálne počítače.